Trabajo de Fin de Grado

Grado en Ingeniería Informática (GEI)

Estudio del uso de memorias no volátiles para mejorar el rendimiento de bases de datos NoSQL

Con la colaboración de Barcelona Supercomputing Center (BSC)

Autor: Enric Sosa Cintero

Directora: Yolanda Becerra Fontal

Especialidad: Ingeniería de Computadores

Convocatoria: Otoño 2019

Facultat d'Enginyeria Informàtica (FIB)

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by UPCommons. Portal del coneixement obert de la UPC

Resum

Els discs són els dispositius d'emmagatzematge amb el temps d'accés més alt. De diverses formes, l'estructura de la jerarquia de memòria d'un sistema informàtic sempre s'ha intentat optimitzar. No obstant, el disc és l'única memòria que manté persistent les dades, és a dir, després de l'apagada de sistema, es mantenen en memòria. Per tant, en la majoria de sistemes informàtics els discs són indispensables. Així doncs, l'accés a ells és inevitable, de manera que millorar un sistema amb la seva interacció amb aquests dispositius és fàcilment beneficiós.

Intel ha proposat una nova tecnologia no volàtil amb un temps d'accés poc superior a les memòries DRAM. Es pretén que la persistència d'aquestes dades no resulti tan costosa com els discs tradicionals HDD o SSD. Per comprovar la seva efectivitat, es posarà a prova en diferents entorns. Un bon exemple de sistema informàtic amb moltes operacions a disc són les bases de dades. En aquest cas, la base de dades serà Cassandra.

Quan una base de dades, com Cassandra, escriu les dades de les taules a disc, les escriu de fila en fila. Apache Arrow, una llibreria d'Apache[1], permet escriure les dades en columnes, aprofitant les característiques SIMD dels nous processadors. Així mateix, el format en què es guarden les dades es pretén que sigui el mateix per a totes les bases de dades. Aquest programari encara és nou i sembla tenir potencial, de manera que aquest treball també aspira a poder fer un anàlisi de la llibreria aplicada en Cassandra amb la memòria no volàtil.

Resumen

Los discos son los dispositivos de almacenamiento con el tiempo de acceso más alto. De diversas formas, la estructura de la jerarquía de memoria de un sistema informático siempre se ha intentado optimizar. Sin embargo, el disco es la única memoria que mantiene persistente los datos, es decir, después de la apagada de sistema, se mantienen. Por lo tanto, en la mayoría de sistemas informáticos los discos son indispensables. Así pues, el acceso a ellos es inevitable, por lo que mejorar un sistema con su interacción con estos dispositivos es fácilmente beneficioso.

Intel ha propuesto una nueva tecnología no volátil con un tiempo de acceso poco superior a las memorias DRAM. Se pretende que la persistencia de estos datos no resulte tan costosa como los discos tradicionales HDD o SSD. Para comprobar su efectividad, se va a poner a prueba en diferentes entornos. Un buen ejemplo de sistema informático con muchos operaciones a disco son las bases de datos. En este caso, la base de datos será Cassandra.

Cuando una base de datos, como Cassandra, escribe los datos de las tablas en memoria, los escribe de fila en fila. Apache Arrow, una librería de Apache[1], permite escribir los datos en columnas, aprovechando las características SIMD de los nuevos procesadores. Asimismo, el formato en que se guardan los datos se pretende que sea el mismo para todas las bases de datos. Este software todavía es nuevo y parece tener potencial, por lo que este trabajo también aspira poder hacer un análisis de la librería aplicada en Cassandra con la memoria no volátil de Intel.

Abstract

Disks are the storage with the highest access time. In various ways, the structure of the memory hierarchy of a computer has always been tried to optimize. However, the disk is the only memory that keeps data persistent, that is, after system shutdown, they are maintained. Therefore, in most computer systems the disks are indispensable. Thus, access to them is inevitable, so improving a system with its interaction with these devices is easily beneficial.

Intel has proposed a new non-volatile technology with a slightly longer access time to DRAM memories. It is intended that the persistence of this data is not expensive as traditional HDD or SSD disks. To verify its effectiveness, it will be tested in different environments. A good example of a computer system with many disks operations are databases. In this case, the database will be Cassandra.

When a database, such as Cassandra, writes the data in memory, it writes them from row to row. Apache Arrow, an Apache library[1], allows you to write the data in columns, taking advantage of the SIMD features of the new processors. Also, the format in which the data is stored is intended to be the same for all databases. This software is still new and seems to have potential, so this work also aims to make an analysis of the library applied in Cassandra with Intel's non-volatile memory.

Índice general

Índice de figuras

Glosario

API Application Programming Interface. 15, 32, 40, 43

BSC Barcelona Supercomputing Center. 12, 15-16, 19-20, 29-30, 51-52, 61

CQL Cassandra Query Language. 18, 44

CPU Central Processing Unit. 40-41, 59

DAX Direct Access Xpoint. 32-33, 39-40, 43

DIMM Dual In-line Memory Module. 29, 60

DRAM Dynamic Random Access Memory. 12-13, 30-31

GEP Gestió de Projectes. 19, 23, 26, 52

GPU Graphics Processing Unit. 40

HDD Hard Disk Drive. 12-13, 32-33, 40, 61, 63

IDE Integrated Development Environment. 20-21

NoSQL Non Structured Query Language. 12, 28-29

PMEM Persistent Memory. 12, 28

SSD Solid-State Drive. 12-13, 32-33, 43, 61

SIMD Single Instruction, Multiple Data. 12, 14, 27, 41-42

TIC Tecnologías de Información y Comunicación. 24

1 Introducción y contextualización

Todos los programas informáticos con gestión de operaciones a disco, como las bases de datos, siempre han sufrido una demora en el rendimiento. Durante su implementación, siempre se han intentado varios métodos para que el tiempo de acceso a disco sea el menor posible. Aún así, es inevitable ya que es la única forma de guardar los datos de forma persistente.

Intel se ha dado cuenta de este problema y pretende solucionarlo. Su propuesta será un tipo de memoria que comparta muchas propiedades de la tecnología DRAM pero, al mismo tiempo, ser no volátil, así pudiendo sustituir el disco tradicional: HDD y SSD.

1.1 Contexto

Este proyecto es un Trabajo de Fin de Grado (TFG) del Grado de Ingeniería Informática (GEI) impartido por la Facultad de Informática de Barcelona (FIB). Este trabajo ha sido desarrollado bajo la tutela de Barcelona Supercomputing Center (BSC), en colaboración con Intel, con la finalidad de aplicar su tecnología de memoria persistente en bases de datos NoSQL.

Esta memoria persistente (PMEM, también llamada Optane) desarrollada por Intel tiene como propiedades poder acceder a ella a nivel de byte, como las DRAM, y ser no volátil, como las SSD o HDD. Además, ofrece una latencia de acceso un poco mayor a la DRAM pero mucho menor que las no volátiles mencionadas antes. Estas propiedades hacen de este dispositivo conveniente para las bases de datos, que realizan muchas operaciones a disco.

Un ejemplo puede ser Cassandra, una base de datos distribuida no relacional. Cassandra tiene opciones de implementación flexibles, altamente eficiente (especialmente para escrituras), escalable y tolerante a fallos. Cassandra es open source, esto es, un modelo de desarrollo de software con exposición del código al público que fomenta la colaboración abierta. Asimismo, es una base de datos muy utilizada en el departamento Computer Science del BSC, contexto donde se encuentra este proyecto. Esto nos ha permitido a que sea un candidato adecuado para nuestro proyecto.

Finalmente, también se va a aplicar la librería Apache Arrow en Cassandra. Apache Arrow nos permite poder guardar los datos en formato columnar. De esta forma, se podrán utilizar las nuevas prestaciones de los procesadores: las instrucciones SIMD. Estos datos también aprovecharemos en guardarlos en el Optane y estudiar si nos hará ganar rendimiento en ciertas lecturas de Cassandra.

1.2 Formulación del problema

En la teoría Intel promete un sistema de almacenamiento persistente muy interesante. En computación, los dispositivos de almacenamiento tradicionales – sea SSD o HDD – son muy lentos en comparación con la DRAM. Ahora bien, resulta ser una tecnología emergente en que todavía no se han hecho públicos muchos resultados concluyentes.

Por otro lado, tenemos las bases de datos, como Cassandra, que sufren del gran coste de acceso a disco. Dependen de realizar muchas transacciones en el menor tiempo posible, así que resulta un punto crítico la gestión de estas transferencias de datos. Por lo tanto, se pretende comparar el rendimiento de Cassandra aplicando la tecnología e implementación de Intel.

Sobre este Cassandra modificado también se le añadirá la prestación de poder guardar los datos en formato columnar utilizando Apache Arrow. Cassandra es una base de datos optimizada con sus escrituras pero sus lecturas resultan lentas. Así pues, dependiendo de la transacción, se va a utilizar los datos columnares para agilizar las lecturas.

1.3 Agentes implicados

En este proyecto habrá de agentes que estarán directamente implicados en su realización, además de los agentes que se beneficiarán y utilizaran los resultados:

- **Barcelona Supercomputing Center** - En concreto, el departamento de Computer Science, el cual trabaja en el diseño de gestión de recursos para aplicaciones de big data. Algunos proyectos de esta línea de investigación actúan sobre Cassandra, así que una mejora del rendimiento de esta base de datos puede beneficiar a dichos proyectos.
- **Intel** - Como fabricantes de la tecnología sobre la cual se va a actuar, resulta obvio su interés en cómo se desenvuelve en distintas aplicaciones. Como se ha dicho anteriormente, esta tecnología tiene pocos años, por lo que desarrollos y resultados factibles les resultará de gran ayuda.
- **Investigador** - El principal responsable de planificar, desarrollar, implementar y documentar el proyecto, así como procesar los resultados y sacar conclusiones.
- **Directora** - La supervisora del investigador, quien va a planificar las tareas por adelantado, para así dirigir y aconsejar el desarrollo del proyecto. Asimismo, resulta ser la jefa del equipo de investigación, por lo que será directamente beneficiada por los resultados.

2 Alcance, metodología y planificación

El Optane de Intel todavía es una tecnología joven, por lo que aún hay carencia de investigaciones públicas. Más aun cuando este producto está dirigido a servidores y no al público general, no todas las entidades son capaces de elaborar estudios sobre su rendimiento.

Por último, el formato columnar también resulta novedoso. Esta metodología se aprovecha de las instrucciones SIMD de los procesadores actuales, propiedad relativamente reciente. Un proyecto que permite implementar esta prestación es Apache Arrow, que todavía no ha llegado a una versión oficial. Públicamente, hay pocas investigaciones pero ya empiezan a surgir tests resultantes de la aplicación a bases de datos.

2.1 Alcance del proyecto

Antes de empezar con el proyecto hay que definir sus objetivos y cómo llegar a ellos. Esto es, la motivación que empuja al proyecto, los requisitos a alcanzar y los posibles obstáculos y riesgos que pueden surgir durante su desarrollo.

2.1.1 Motivación y objetivos

Como se ha expuesto, una buena implementación de la tecnología de Intel en Cassandra tiene gran potencial de mejora. Todas las innumerables transacciones que se van a realizar en esta base de datos pueden llegar a tener un coste de tiempo menor. Además, si se reduce el tiempo de ejecución de las operaciones a disco, permitirá un mayor ancho de banda. Con todo, esta versión modificada de Cassandra podría encajar con el resto de proyectos del departamento y la misma metodología puede ser aplicada a otras bases de datos. Finalmente, sobre esta memoria, también se van a guardar los datos en formato columnar paralelamente, esperando una mejora en algunas lecturas que aprovechen esta disposición de los datos. La idea es que la ganancia en estas lecturas sea muy grande para que valga la pena el código extra que requieren las escrituras.

2.1.2 Requisitos del proyecto

Para que los objetivos se cumplan y poder solventar los problemas planteados, hay que definir los requisitos que el proyecto debería cumplir. De esta forma, junto con las metodologías definidas posteriormente, nos va a permitir agilizar el progreso del proyecto. A continuación están los requisitos que se deberán cumplir:

- Reducir el tiempo de ejecución de Cassandra.
- Que la réplica de los datos en formato columnar, no consuma más tiempo y memoria de lo necesario.
- Tener un uso eficiente del consumo de memoria del Optane de Intel.
- Ser lo más transparente al usuario lo más posible.
- Programar teniendo en cuenta l'arquitectura de la máquina Apache Pass del BSC, la cual tiene implementados los módulos de memoria persistente de Intel.
- Emplear buenas prácticas de programación, con un estilo legible y la menor complejidad posible.

2.1.3 Posibles obstáculos

Ocupación de la máquina Apache Pass

Todas las pruebas de rendimiento se realizarán en la máquina Apache Pass del BSC. En este sistema hay otros usuarios que aprovechan el Optane de Intel para sus propios proyectos. Además, no hay un sistema de colas. En consecuencia, esto puede provocar diversas interferencias durante la implementación de mi proyecto.

Solución: Durante la implementación y verificación del código no hay problema porque se puede trabajar utilizando una máquina individual. Sin embargo, las medidas de rendimiento que se van a extraer será un tema más delicado. Cuando proceda, se tendrá que pedir la exclusividad de Apache Pass y configurar ciertos parámetros según este entorno.

Fallos de programación

Reiterando que se está trabajando sobre una tecnología relativamente nueva, también lo son las librerías disponibles que permiten operar sobre esta persistencia de memoria.

Solución: Un estudio del funcionamiento básico de estas librerías, en contexto con el código que se va a trabajar (Java, en este caso). Esto también implica leerse las APIs que ofrecen estas librerías y leerse su documentación.

Código de Cassandra

Cassandra tiene casi 400.000 líneas de código [2]. A pesar de que, conceptualmente, Cassandra es fácil de entender y está bien documentado, no resulta tan fácil localizar donde modificar exactamente el código para que a) haga lo que tenga hacer; b) no interfiera con el resto de prestaciones.

Solución: Estudiar su documentación analizando el código. Se deberán de hacer pruebas antes del código para persistencia de memoria para asegurarse dónde empezar a modificar la implementación del Cassandra.

Apache Arrow en desarrollo

En los estos momentos que se está escribiendo este documento, la versión más reciente de esta herramienta es la 0.14.1. Todavía está en unas fases tempranas de desarrollo, por lo que pueden haber bugs y no puede estar muy bien optimizado.

Solución: Habrá que prestar atención en el desarrollo de esta aplicación. Es muy probable que durante este TFG, Apache Arrow avance en sus versiones. Así pues, se irán consultando todas las notificaciones de su progreso y sopesar si vale la pena actualizar la implementación.

2.1.4 Riesgos críticos

Virus / error fatal en portátil

En cualquier entorno informático puede haber una corrupción crítica en la memoria, es decir, en los datos. Sea por estar en conexión con internet o bien por inexperiencia con instalaciones de las librerías y softwares, un virus o una caída inesperada de sistema puede tirar por la borda varias horas de avance.

Solución: Destacar que el portátil en sí no es vital ya que los tests se van a realizar en la máquina Apache Pass. La inhibición del portátil sería un contratiempo pero el desarrollo puede seguir avanzando en otro ordenador. Por lo que se refiere al trabajo guardado en el portátil, siempre va a ver una copia en la nube. Específicamente, GitHub y Google Drive para el código y la documentación, respectivamente.

Error crítico en la máquina Apache Pass

Esta máquina del BSC está en servicio para varios usuarios. Muchas veces cada uno realiza tests los cuales necesitan propiedades hardware distintos (más RAM, etc.). Por lo tanto, es una máquina en la que se hacen muchos cambios y reinicios. Esto puede llevar contratiempos y posibles malversaciones del hardware.

Solución: De forma fácil (mediante Slack) se puede pedir a los administradores de la máquina resetearla con las propiedades que uno necesita y que se mantenga hasta el fin de todas las ejecuciones correspondientes. Por otro lado tenemos el caso de si se estropea, en especial el hardware de Intel. En tal caso, realmente resulta grave pero, a pesar de no disponer de la tecnología, se puede virtualizar y realizar tests sobre esta tecnología simulada.

Obviamente, requeriría mucha inversión de tiempo y trabajo pero se podría evitar un punto crítico de no retorno.

2.2 Metodología y rigor

Va a ser necesario definir el conjunto de métodos que se van a utilizar durante la investigación para consolidar un desarrollo estable y factible. Esto es, los diversos principios empleados durante el avance, las herramientas utilizadas como control de seguimiento y los elementos de verificación de los resultados.

2.2.1 Métodos de trabajo

Durante todo el desarrollo del proyecto se seguirá una forma de trabajar iterativa. Es decir, primero se analizará el estado del proyecto, luego se planificarán las posibles mejoras, se implementarán y finalmente se validará el progreso realizado. Entonces, esta metodología se volverá a repetir hasta conseguir los objetivos finales del proyecto. Cada una de estos pasos se documentará para facilitar posteriores iteraciones.

Asimismo, durante la implementación del código, se valorarán juegos de prueba que validen los casos más excepcionales que pueden ser dados. Esto ayudará para tener en cuenta los detalles que cubren estos casos, y así robustecer el código antes de encontrar problemas.

Esta forma de trabajar se verá consolidada con una constante supervisión por parte de la directora del proyecto. Durante el trabajo, se intercambiarán ideas para el avance del progreso y se expondrá antes de continuar para poder mejorar y corregir errores. Además, otros compañeros del departamento también podrán consultar el desarrollo del proyecto.

2.2.2 Herramientas de seguimiento

Como herramienta para control de versiones va a ser GitHub. Esto nos permitirá facilitar la disponibilidad del código y la recuperación de versiones anteriores en caso de fallos críticos. Se utilizarán dos repositorios para el mismo proyecto. Uno va a ser personal para el investigador, en el cual habrá comúnmente iteraciones a medio hacer de desarrollo y debugging. El otro será visible para la directora y otros compañeros de trabajo, para facilitar el seguimiento del proyecto. En este último sólo se constatarán las versiones más estables del desarrollo.

2.2.3 Métodos de validación

Se programarán reuniones presenciales semanales, en que participarán investigador, directora y otros compañeros para validar el progreso del proyecto. En estas reuniones se podrá discutir sobre el estado del proyecto.

Como gestor de validación de las capacidades técnicas del proyecto, podremos utilizar diversas aplicaciones para interactuar con Cassandra. Con Nodetool [3] nos permitirá gestionar un cluster de Cassandra mediante comandos. Con cqlsh [4] podremos conectarnos a un cluster y enviarle comandos CQL. Estas herramientas nos servirán para controlar y gestionar diversas operaciones en Cassandra y sacar medidas. Por último dispondremos de librerías de Java que ayudarán a sacar tiempos y calcular el rendimiento de trozos de código en específico.

2.2.4 Metodología y rigor final

Durante este subapartado, se ha expuesto la metodología que se seguiría durante el desarrollo del proyecto. Aquí se pretende exponer las estrategias tomadas, justificándolas para corroborar el progreso del trabajo. Asimismo, se valorará como se ha intentado evitar los riesgos anteriormente explicados.

Los métodos de trabajo vienen comentados en **2.2.1 Métodos de trabajo**. Resumiendo, se trata de trabajar de forma iterativa con constantes pruebas para validar el progreso y supervisión de la directora, si puede ser en persona para habilitar mejor comunicación. En los siguientes capítulos se definen las tareas a seguir, pero por encima se van a dar ahora, para poder justificar los métodos de trabajo aplicados en ellas.

Durante el diseño de la implementación, se ha necesitado ayuda de los desarrolladores de software más expertos del departamento. Aquí se ha intentado tomar mucha precaución, porque gran parte del desarrollo y tiempo invertido se partirá del diseño tomado previamente. Así pues, para esta fase sólo se ha necesitado de una iteración, pero se ha planteado con sumo cuidado. Por otro lado, a pesar de que no se haya producido de forma iterativa, la reunión sí que fue presencial con tal de que los conceptos quedaran claros.

Entonces, durante la implementación sí que se ha hecho de forma iterativa. Es costumbre que, en entornos informáticos, si no se prueban las cosas muy de seguido, surjan muchos problemas. Además, teniendo en cuenta que se está partiendo del código enorme de Cassandra, era indispensable tener que hacer pruebas de seguido con tal de no romper el programa. Otra razón de más es que el investigador ha tenido más libertad durante la implementación que durante el diseño previo; por lo que más errores podrían ocurrir. Así

pues, durante esta fase sí que se ha necesitado de muchas iteraciones y supervisiones de la directora con tal de sacar adelante el proyecto.

Por otro lado, tenemos las pruebas para verificar el código durante su implementación. Se ha visto que esta metodología no ha sido realmente necesaria, ya que muchas prestaciones se han pasado por alto con tal de poder acabar con el proyecto y así tener el esqueleto principal de la implementación. Es decir, pruebas exhaustivas de validación no se han realizado, pero sí que han sido necesarias algunas más superficiales con tal de verificar la efectividad de lo codificado.

Por último, falta detallar cómo se han evitado los riesgos críticos detallados en **2.1.4 Riesgos críticos**. Primero de todo empezar por el error crítico en la máquina Apache Pass, porque el riesgo no es responsabilidad nuestra. Aún así, es de esperar que la probabilidad de que pase sea muy baja, porque en la máquina trabajan varios departamentos del BSC y se verían afectados. Además, en teoría está gestionado por expertos. El otro problema era si se estropea el ordenador portátil proporcionado por el BSC. En este ordenador, sólo se ha intentado descargar e instalar lo necesario, con tal de minimizar las probabilidades de virus y falladas de sistema. Además, siempre que se ha movilizado, se ha intentado llevarlo siempre protegido dentro de una mochila con tal de evitar daños físicos. Durante todo el desarrollo no han sucedido ninguno de estos problemas, por lo que el progreso del proyecto sólo ha sido dependiente del trabajo del grupo de trabajo, principalmente del investigador.

2.3 Planificación temporal

El tiempo aproximado de este proyecto es de cuatro meses, desde principios de septiembre hasta finales de enero, acabando con la defensa del proyecto. En este apartado se detallarán la tareas que conglomeran el avance del proyecto, junto con el tiempo aproximado que requerirán y los posibles obstáculos que se podrían encontrar. Finalmente, se comparará la planificación inicial con la final resultante, y se describirán las causas de las diferencias.

2.3.1 Descripción de las tareas

Gestión del proyecto

Esta tarea corresponde al curso GEP, vinculado al trabajo de fin de grado de la FIB. El
objetivo del curso es aprender a hacer una planificación del trabajo a cumplir durante el proyecto. El contenido de esta parte es:

- Herramientas TIC de soporte a la gestión de proyectos y equipos

- Contextualización y alcance
- Planificacion temporal
- Presupuesto y sostenibilidad
- Evaluación final

Destacar que en todas las tareas es necesario un ordenador con conexión a Internet. Asimismo, en esta es necesario Google Drive para escribir los entregables, un Gantter y el Racó y Atenea.

Análisis y diseño

Los principales elementos del proyecto son Cassandra, el Optane de Intel y Apache Arrow. Antes de la implementación, primero habrá que estudiarlos. En esta etapa se tratará de familiarizarse con todos los conceptos que circundan estos elementos. También se leerá la documentación de la máquina sobre la cual se va a trabajar con tal de hacerse con sus características.

Luego del análisis, toca el diseño. Como todo proyecto que interviene software, es más importante el diseño hecho previamente que la implementación. Aquí se definirán las relaciones entre los diversos elementos implicados y el esquema general de la implementación.

En esta tarea es necesario un IDE que facilite la gestión y navegación del código así como la máquina Apache Pass del BSC. Consecuentemente, todas las siguientes tareas que también impliquen código y su ejecución, necesitarán de los recursos anteriores. Asimismo, serán necesarios todos los elementos software que habilitarán las instalaciones de Cassandra y las librerías de Intel y Apache Arrow. Finalmente, para mantener contacto con la directora y el equipo, se va a utilizar Slack como herramienta de comunicación.

Implementación del Cassandra persistente

Tal como se definió en la metodología de trabajo, el proceso de implementación va a ser iterativa. Esta parte va a resultar crítica y hay que asegurarse que se modifica sólo y necesariamente la parte de código de Cassandra necesaria. Esto es, el nivel de riesgo de obstáculos va a ser grande aunque la cantidad de código añadida no sea tan alta.

Como todo proceso de implementación va a ver, la propia implementación, los tests de validación y los tests de rendimiento. En los tests de validación se va a comprobar que el

código es correcto y pasa todos los juegos de prueba. Por último, los tests de rendimiento ayudarán a consolidar la dirección tomada.

Durante la implementación, los recursos necesarios van a ser similares a los utilizados durante el análisis y diseño. Simplemente destacar que, con que los tests requieren de un entorno delicado, la reserva exclusiva de la máquina Apache Pass también se va a pedir por Slack. Para documentar los resultados, se utilizará Google Drive y LibreOffice.

Implementación de Apache Arrow

La metodología de esta parte es muy parecida a la anterior. Esta parte se prevé que va a tener menos riesgos que la anterior ya que, una vez se sabe dónde intervenir en el código de Cassandra, la implementación de este código va a ser más fácil. Aún así, hay que tener en cuenta que es una librería joven en constante actualización, por lo que esto puede causar demoras. Los recursos van a ser los mismos que los utilizados en el apartado anterior.

Evaluación de rendimiento

Resulta importante saber qué partes del código mejoran o empeoran el rendimiento. Principalmente, tenemos dos elementos que medir: la memoria persistente y el formato columnar. Esto causa que esta evaluación sea más densa de lo normal, ya que los tests nos deberían dar información clara sobre qué elemento afecta al rendimiento y en qué grado.

Asimismo, esta parte también puede implicar hacer modificaciones en el código según los resultados sacados. Así pues, esto también añadirá tiempo que invertir con tal de conseguir el mejor rendimiento posible. Los recursos van a ser los mismos que los utilizados en los apartados de implementación.

Hito final del proyecto

Por último, se documentará el proyecto una vez finalizado. Una vez hecha la memoria, sólo queda preparar la presentación oral ante un tribunal de la especialidad. Será uso del IDE Overleaf para la documentación de la memoria y las diapositivas para la presentación oral.

2.3.2 Estimación temporal

Una vez definidas las tareas y sus subtareas, toca estimar el tiempo que van a consumir y precisar sus dependencias. La siguiente tabla va a resumir estos detalles. Posteriormente, en un diagrama de Gantt se va a visualizar mejor esta estimación temporal.

Id.	Descripción	Tiempo aprox.	Dependencias
T1	**Gestión del proyecto**	**75h**	
T2	→ Herramientas TIC de soporte	14,75h	-
T3	→ Contextualización y alcance	24,50h	-
T4	→ Planificación temporal	8,25h	T3
T5	→ Presupuesto y sostenibilidad	9,25h	T4
T6	→ Evaluación final	18,25h	T5
T7	**Análisis y diseño**	**60h**	
T8	→ Análisis de los elementos	48h	-
T9	→ Diseño de la implementación	12h	T8
T10	**Cassandra persistente**	**120h**	
T11	→ Implementación del código	80h	T9
T12	→ Tests de validación	20h	-
T13	→ Tests de rendimiento	20h	-
T14	**Apache Arrow**	**100h**	
T15	→ Implementación del código	60h	T13
T16	→ Tests de validación	20h	-
T17	→ Tests de rendimiento	20h	-
T18	**Evaluación de rendimiento**	**40h**	**T17**
T19	**Hito final**	**75h**	
T20	→ Documentación del proyecto	65h	-
T21	→ Preparación de la defensa	10h	T20

Fig. 1 Tabla de las tareas con su duración y dependencias

Se estima una duración total de unas 470 horas. A continuación se muestra un Gantt para visualizar mejor las tareas dentro del marco temporal del desarrollo del proyecto:

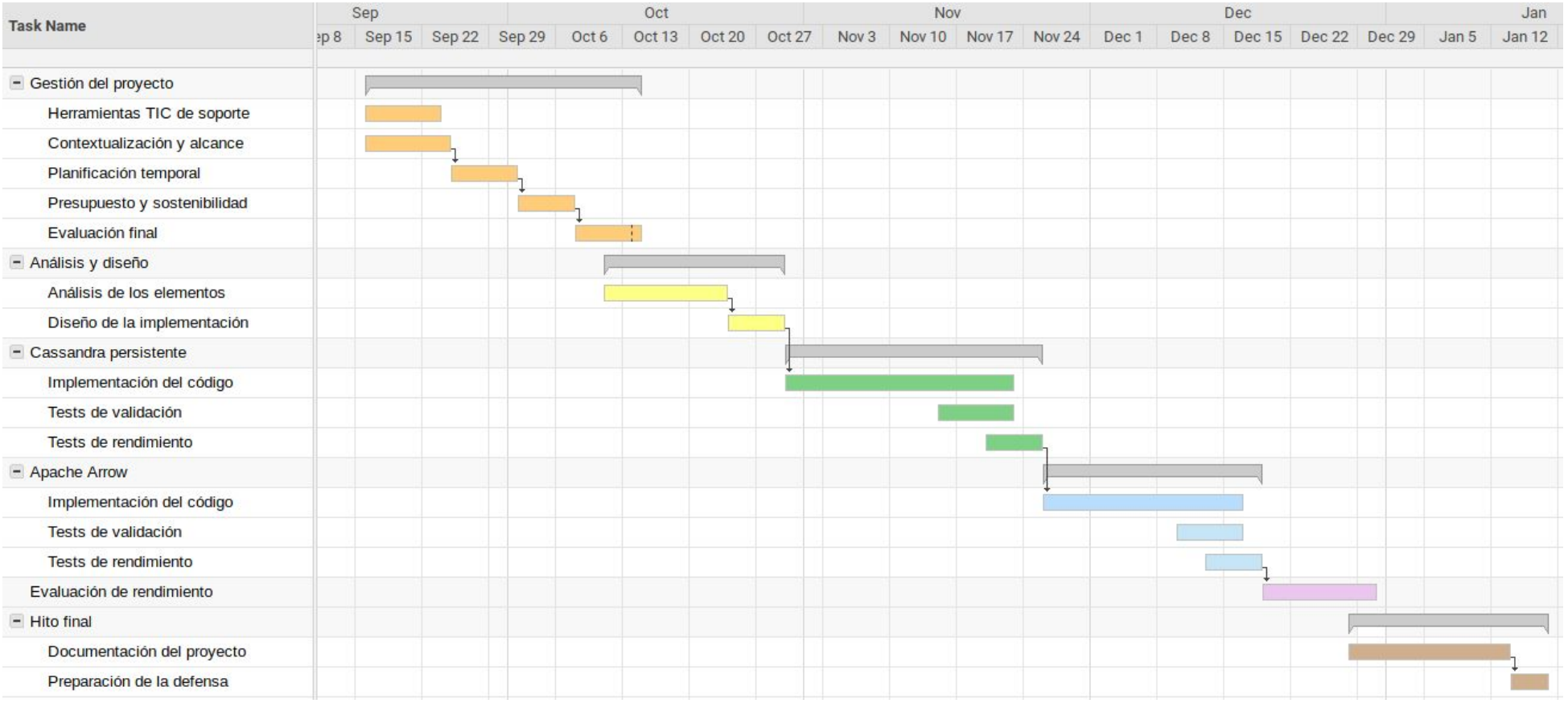

Fig. 2 Diagrama de Gantt durante el hito inicial

2.3.3 Planificación final

Aquí se presenta la planificación final del proyecto, es decir, el resultado de la planificación inicial y las alteraciones del proyecto durante su desarrollo. La tabla siguiente (Figura 3) se definen las tareas especificadas anteriormente y se detalla alguna observación relacionada con su ejecución. Posteriormente, se precisa alguna observación y su impacto en el progreso del trabajo. Finalmente, se muestra el diagrama de Gantt actualizado en que se visualiza mejor el estado del proyecto dentro de su marco temporal.

Tarea	Descripción	Estado	Observaciones
T1	**Gestión del proyecto**	**Completado**	
T2	→ Herramientas TIC de soporte	Completado	-
T3	→ Contextualización y alcance	Completado	-
T4	→ Planificación temporal	Completado	-
T5	→ Presupuesto y sostenibilidad	Completado	-
T6	→ Evaluación final	Completado	-
T7	**Análisis y diseño**	**Completado**	
T8	→ Análisis de los elementos	Completado	-
T9	→ Diseño de la implementación	Completado	-
T10	**Cassandra persistente**	**Completado**	
T11	→ Implementación del código	Completado	Intel ya inició su proyecto en modificar Cassandra incluyendo su persistencia de memoria. Este código es open source y se ha partido de él.
T12	→ Tests de validación	Completado	-
T13	→ Tests de rendimiento	Completado	No ha sido necesario definir los tests porque la herramienta cassandra-stress[5] nos permite ejecutar operaciones ágilmente
T14	**Apache Arrow**	**Completado**	
T15	→ Implementación del código	Completado	-
T16	→ Tests de validación	Completado	Estos tests se han implementado en C++. Originalmente iban a estar en Java, pero se ha concluido que en C va a ser más eficiente.
T17	→ Tests de rendimiento	Completado	Las mismas observaciones que la tarea anterior.
T18	**Evaluación de rendimiento**	**Completado**	-
T19	**Hito final**	**En desarrollo**	
T20	→ Documentación del proyecto	En desarrollo	-
T21	→ Preparación de la defensa	Pendiente	-

Fig. 3 Tabla de las tareas con su duración y dependencias

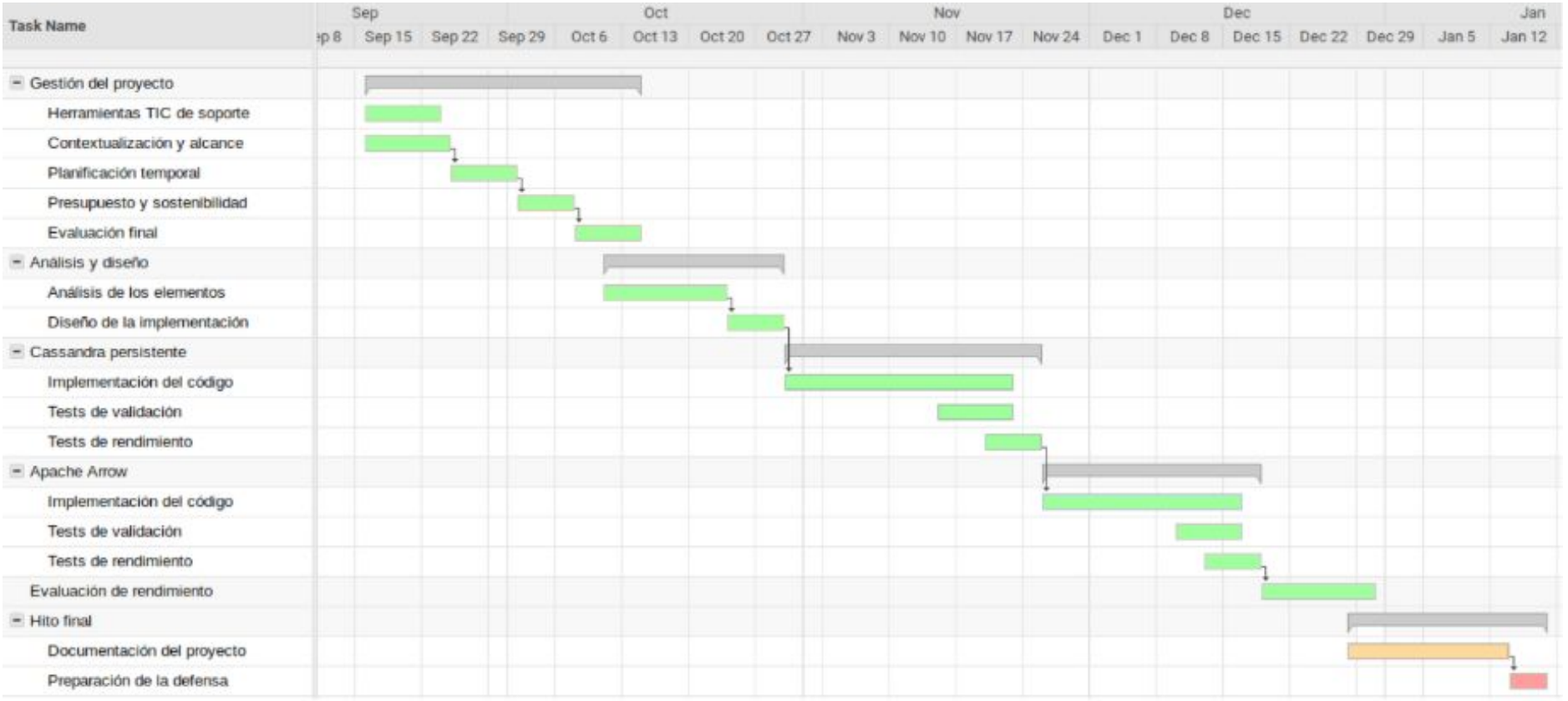

Fig. 4 Diagrama de Gantt final. En verde las tareas completadas, en naranja las que están en desarrollo y en rojo las que todavía están pendientes.

Durante el hito inicial, en GEP, se habían definido diversos problemas y riesgos que podrían surgir y, en base a estos, proponer sus soluciones y alternativas. Ahora, acabando el desarrollo del proyecto, se ha tenido la suerte de no haber recurrido a ninguna de estas alternativas. A causa de que en su día se definió adecuadamente la estructura del proyecto, su avance todavía no ha visto inconvenientes. Sin embargo, algunas especificaciones del código del proyecto sí que se han visto alteradas. Esto es el caso de los tests de la implementación de Apache Arrow.

El lenguaje del código de Arrow se ha decidido cambiarlo. En concreto, la parte encargada de realizar las lecturas a Cassandra. Estas lecturas nos sirven para verificar los datos escritos en Arrow y recoger tiempos y resultados.

Originalmente, el código estaba en Java, ya que se reciclaría código de Arrow utilizado en Cassandra para las escrituras. Pero, con que Arrow es una librería cross-language, se ha barajado la posibilidad de implementarlo en C++. El motivo principal es que, generalmente, C++ va a resultar más óptimo, por lo que los resultados sacados a partir de este código van a ser más veraces. La implementación en C ha causado más problemas de lo normal. Además, otro motivo es que estos datos de Arrow están guardados en el Optane y, hasta ahora, se había utilizado una librería en Java. Esto último también ha complicado la interacción con la persistencia de memoria por haber cambiado de lenguaje.

A pesar de haber costado más de lo esperado, el coste calculado durante el hito inicial no se ve perjudicado. En el subapartado **6.7. Desviaciones y presupuesto final**, se detalla la relación entre el presupuesto total calculado al principio con el final.

3 Estado del arte

El Optane de Intel todavía es una tecnología joven, por lo que aún hay carencia de investigaciones públicas. Más aun cuando este producto está dirigido a servidores y no al público general, no todas las entidades son capaces de elaborar estudios sobre su rendimiento.

Con que pretende solventar la lentitud de dispositivos de almacenamiento tradicionales, sus primeras implementaciones han sido en entornos donde se producen muchas operaciones a disco, como por ejemplo, sistemas de fichero o bases de datos. Así pues, a pesar de su poca edad, podemos ver ejemplos aplicados a sistemas de ficheros[6] o bases de datos[7].

Por otro lado, Cassandra tiene más historia[8]. El funcionamiento y metodología de esta base de datos está extensamente documentada[9], por lo que fácilmente se puede informarse de sus conceptos. Cassandra nació de un proyecto de Facebook donde posteriormente acabaría siendo un proyecto open source. Actualmente, Cassandra es utilizado por eBay, GitHub, Instagram, etc[10].

Por último, el formato columnar también resulta novedoso. Esta metodología se aprovecha de las instrucciones SIMD de los procesadores actuales, propiedad relativamente reciente. Un proyecto que permite implementar esta prestación es Apache Arrow[1], que todavía no ha llegado a una versión oficial. Públicamente, hay pocas investigaciones pero ya empiezan a surgir tests resultantes de la aplicación a bases de datos[11].

3.1 Punto de partida

En un principio cabría esperar que el proyecto partiría de la implementación original de Cassandra, pero Intel ya ha empezado con su propuesta. En el repositorio siguiente[12] se encuentra la versión de Cassandra 4.0 modificada por ingenieros de Intel con tal de que todas las operaciones de lectura y escritura a disco se hagan en su Optane.

Así pues, tenemos una versión de Cassandra persistente hecha por Intel. Este proyecto partiría de esta implementación, el problema es que es un código relativamente reciente (14 de enero de 2019) y poco documentado. Por lo tanto, a pesar del indudable estatus de los ingenieros de Intel, se tendrán que estudiar los cambios hechos por ellos y cambiar los trozos de código con tal de que funcione en nuestro entorno.

Por lo que se refiere a la implementación de Arrow, todavía no hay documentos ni código con su implementación en Cassandra, por lo que el punto de partida será cuando se haya acabado de moldear el Cassandra persistente.

4 Cassandra con memorias no volátiles

En este capítulo se detallan las características del Optane de Intel utilizado y la máquina en el que se ha instalado. A continuación, se describe el desarrollo de la implementación: introducción y diseño; implementación y detalles del código y finalmente tests, resultados y conclusiones.

Durante este proceso se han determinado dos etapas: el Cassandra persistente, el cual trata del Cassandra que lee y escribe de las tablas guardadas en PMEM, y el Cassandra con Arrow, sobre el Cassandra persistente anterior, en que se van a añadir tablas con datos en formato columnar.

Con estas dos nuevas prestaciones se pretende estudiar las nuevas tecnologías no volátiles aplicadas en bases de datos NoSQL. Además, sobre esta tecnología, también se va a investigar el uso de datos en formato columnar.

4.1 Entorno de pruebas

Aunque no sea el centro de atención de esta investigación, Cassandra toma un rol importante en este trabajo. Asimismo, Apache Pass, la máquina que tiene el Optane y donde se van a ejecutar y medir todos los tests, también es relevante. Estos componentes formarán el núcleo del entorno de pruebas del código a implementar y de los estudios a sacar.

Para contextualizar el proyecto hay que definir detalladamente dichos elementos. Así pues, este apartado resume sus características y pretende dar una idea del entorno de pruebas en el que se va a someter los elementos principales del estudio: el Optane de Intel y Apache Arrow. Definir Cassandra nos ayudará a entender cómo y cuando la base de datos realiza las lecturas y escrituras. Precisar las características del hardware permitirá evaluar los resultados con mejor criterio y evitar conclusiones erróneas.

4.1.1 La base de datos NoSQL: Cassandra

Cassandra es la base de datos en la que se va a implementar las prestaciones que queremos estudiar. Anteriormente se ha justificado el motivo por el que se ha escogido esta base de datos. Aquí se va a contextualizar Cassandra brevemente explicando un poco su historia y lo que nos permite hacer. Por encima se explicarán sus características para acotar mejor nuestra visión del entorno de la investigación. En capítulos posteriores se van a revelar sus peculiaridades según su importancia en el desarrollo de este trabajo.

Apache Cassandra es una base de datos de código abierto no relacional, o NoSQL, que permite disponibilidad continua, una escalabilidad enorme y una distribución de datos capaz de ser almacenada en múltiples clusters y zonas de disponibilidad en la nube. Asimismo, proporciona una alta velocidad en sus escrituras. En pocas palabras, Cassandra proporciona un motor de almacenamiento de datos altamente confiable para aplicaciones que requieren de una escala muy grande.

Cassandra se desarrolló originalmente en Facebook en 2008, la compañía la lanzó como un proyecto de código abierto en Google Code. En 2010, se convirtió en un proyecto Apache de alto nivel.

Esta versión de código abierto de la base de datos de Cassandra es utilizada por algunas de las compañías de tecnología más grandes del mundo para ejecutar aplicaciones de gran escala. Es ampliamente conocida por su despliegue en Apple. Asimismo, Netflix también es un gran usuario del código abierto de Cassandra.

4.1.2 Entorno hardware: Apache Pass

Apache Pass es un clúster del BSC con 2 nodos de cálculo en el que se ejecutarán las pruebas de rendimiento. Cada nodo contiene 1 socket, en que cada socket contiene un procesador Intel Xeon Platinum 8260L de 24 cores (48 cores en total). Cada nodo tiene 6 DIMMs Intel(R) Optane(TM) DC persistent memory de 514624 MB cada uno (12 DIMMs en total, capacidad de casi 6 TB). El clúster tiene el kernel GNU/Linux 4.18.8 con Fedora v27 como sistema operativo. La librería Apache Arrow es la versión 0.15.1, la más reciente durante esta redacción.

4.2 Cassandra persistente

El primer problema a solucionar de este proyecto es adaptar Cassandra con la memoria no volátil de Intel. Principalmente, se estudiarán las operaciones básicas que hace una base de datos al disco: lecturas y escrituras. Por ello, hay que entender Cassandra y el uso del Optane para poder modificar la bases de datos como se corresponda. Entonces, se modificará el código como es debido y se adaptará a la máquina en que se va a ejecutar: Apache Pass. Finalmente, se someterá a unos tests para poder analizar los resultados y sacar conclusiones.

4.2.1 Introducción

En este apartado se va a introducir la implementación de la memoria no volátil de Intel en Cassandra. Como se ha expuesto anteriormente, Cassandra es una base de datos con un código de más de 100.000 líneas de código. En este trabajo no es esencial entender en

profundidad su funcionamiento e implementacion. Principalmente se ha escogido Cassandra como entorno de pruebas porque es de código abierto, es de interés dentro del grupo de trabajo del BSC y sus lecturas lentas dan margen de mejora.

En el capítulo **3.1 Punto de partida** también se ha justificado otro motivo por el cual se ha escogido Cassandra: Intel ya empezó su implementación del Cassandra persistente[12]. Esto ha resultado de enorme ayuda porque, una cosa es entender el funcionamiento de Cassandra - lo cual no es muy complicado -, y otra muy distinta es ubicar todas estas funcionalidades en su vasto código. Por razones de tiempo, conveniencia y conocimiento, no podíamos determinar la raíz por el cual empezar a modificar código. Así pues, esta versión ya modificada de Intel nos ha sido de gran ayuda para empezar a ajustar el Cassandra persistente para las pruebas que querremos someterle y la máquina Apache Pass.

A pesar de que se parta de código ajeno, habrá que entenderlo y compararlo con el Cassandra original. Por ello, se describirán cómo Cassandra realiza algunas de las prestaciones que principalmente analizaremos: sus lecturas y escrituras a disco. Además, el Optane de Intel permite varios usos, por lo que se justificará el método de empleo de este hardware.

4.2.2 Intel's Optane Memory

En el capítulo **4.1.2. Entorno hardware: Apache Pass** se han descrito las características del Optane utilizado en la máquina del BSC Apache Pass. Sin embargo, a pesar de sus capacidades físicas, este hardware permite un uso muy flexible. Esto es, se puede utilizar de diversos modos y se puede montar en el sistema operativo de distintas maneras. La instalación del dispositivo no es relevante en este trabajo y el proceso es llevado a cabo por parte de otros empleados del BSC. Sin embargo, nos interesa estudiar las capacidades de uso que tiene esta memoria no volátil para escoger el mejor uso para nuestro estudio.

Esta memoria persistente permite dos modos operativos diferentes. Los modos determinan qué capacidades de la memoria están activas y disponibles para el software. Específicamente, cambia la jerarquía de memoria informática convencional según su modo. Esto es crítico por parte del software porque el almacenamiento de sus datos puede verse alterado.

El primer modo es el modo de memoria (memory mode). En este modo, los módulos del dispositivo actúan como módulos de memoria DDR4 de gran capacidad. En dicha configuración, las aplicaciones y el sistema operativo perciben un conjunto de memoria volátil, no de manera diferente a los sistemas con sólo DRAM. En este modo no se requiere una programación de memoria persistente específica en las aplicaciones, y los datos no se almacenarán en caso de apagada del sistema. En el modo de memoria, el Optane actúa como una cache enorme ubicada - dentro de la jerarquía de memoria - entre la DRAM y disco.

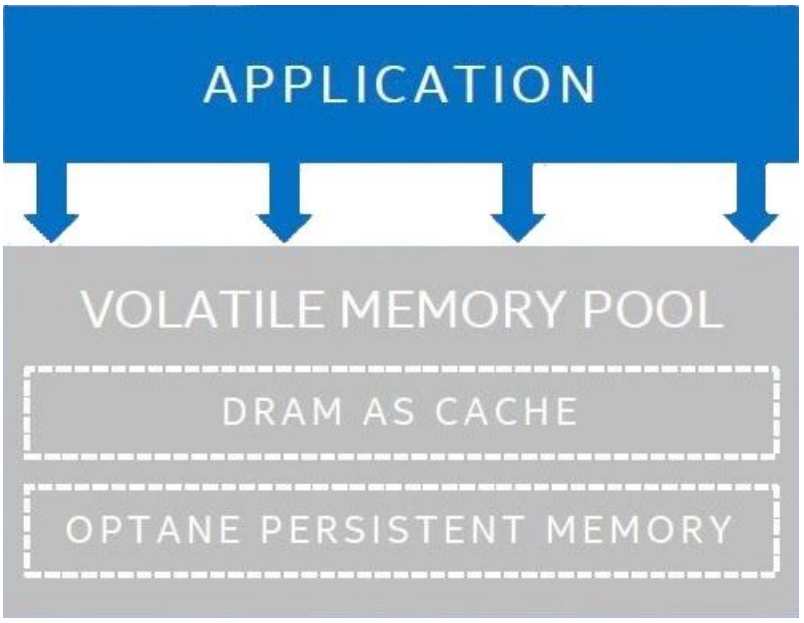

Fig. 5 Esquema de la jerarquía de memoria con Optane de Intel en *Memory Mode*

El segundo modo es el modo App Direct. En este modo, los módulos proporcionan todas las características de persistencia para el sistema operativo y las aplicaciones que los admitan. Configurado en este modo, el software es explícitamente consciente de que hay dos tipos de memoria, y pueden dirigir qué datos son almacenados en DRAM o adecuados para el Optane. Las operaciones que requieren la latencia más baja y no necesiten almacenamiento de datos permanente se pueden ejecutar en DRAM. Ahora bien, los datos que deben ser persistentes o estructuras muy grandes pueden ser dirigidos al Optane de Intel.

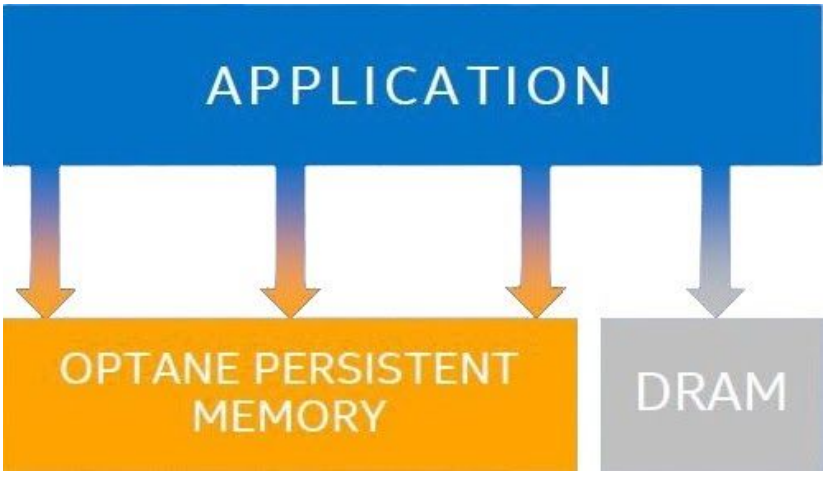

Fig. 6 Esquema de la jerarquía de memoria con Optane de Intel en *App Direct*

Bases de datos en memoria, frameworks en memoria y aplicaciones de almacenamiento rápido de datos persistentes son buenos ejemplos de carga de trabajo que se benefician enormemente del modo App Direct. Es por ello que este proyecto va a trabajar con

este modo, puesto que se va a poner a prueba la gestión de una base de datos para los experimentos.

Dependiendo de la configuración y el sistema operativo, los módulos persistentes pueden montarse en uno de los dos tipos de namespaces:

- **Acceso directo** (direct access, DAX): almacenamiento direccionable a nivel de byte accesible a través de una API. Actúa como un dispositivo de carácter.

- **Almacenamiento de bloque** (block storage): la memoria persistente que se presenta a las aplicaciones se ve como un dispositivo de almacenamiento en bloque, como un SSD o HDD. Actúa como un dispositivo de bloque.

Si se cumple una buena gestión, el modo DAX puede llegar a ser más rápido por el hecho de poder saltarse gran parte del sistema de ficheros del sistema operativo. En el departamento ya se estudió el almacenamiento en bloque y dio resultados muy aproximados respecto las SSD; así que ya se me asignó adaptar Cassandra en acceso directo.

4.2.3 Diseño e implementación

Antes de adentrarnos en la implementación de Intel del Cassandra persistente, se esquematizarán sus escrituras y lecturas para visualizar mejor el contexto.

Cuando hay datos que escribir en memoria no volátil, antes Cassandra los guarda en una estructura de memoria llamada memtable. Las operaciones a disco son muy costosas, por lo que muchas bases de datos son diseñadas con el criterio de accederlo las menos veces posible. Cassandra no es una excepción, la memtable actúa como una cache en memoria con política de escritura write-back, es decir, puede que haya datos que no estén a disco. Cuando por fin Cassandra considera escribir los datos a disco, se dice que ha hecho flush de los datos de las memtables. Las estructuras de datos en disco se llaman SSTables.

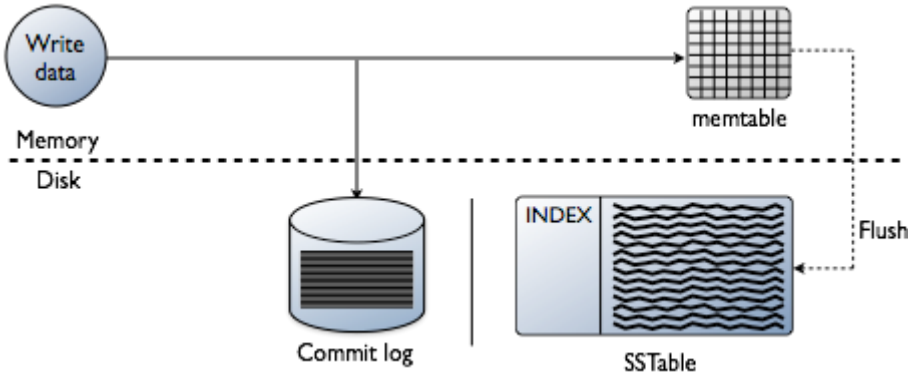

Fig. 7 Esquema del camino de las escrituras en Cassandra

Las lecturas son un poco más complicadas. A causa de la existencia de las memtables y la estructura general de Cassandra, el recorrido de una operación de lectura es más complicada que la de escritura. Pasando por alto los detalles, visibles en la Figura 8, Cassandra empieza a buscar los datos en las memtables acabando por las SSTables.

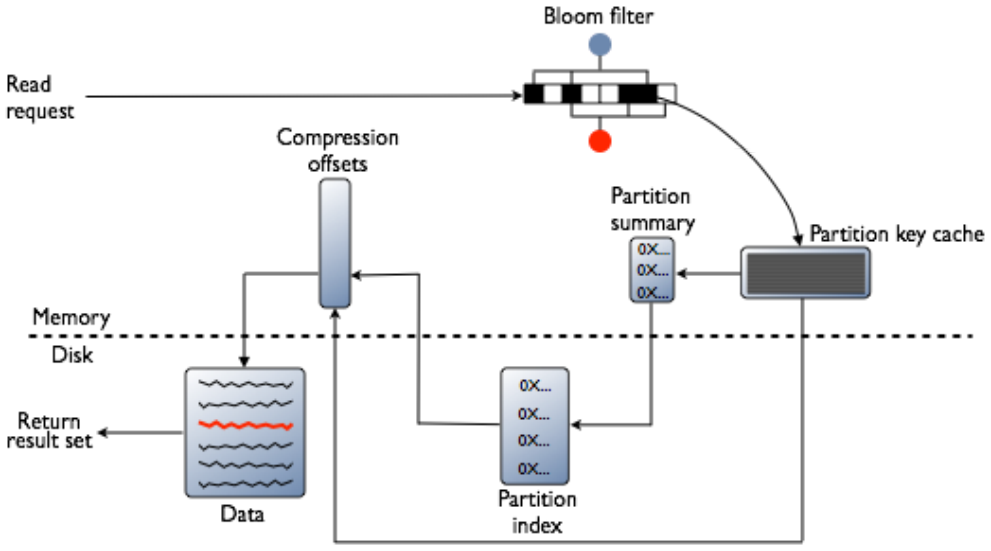

Fig. 8 Esquema del camino de las lecturas en Cassandra

Sobre estas estructuras, Intel ha modificado Cassandra adaptándolo a su hardware en App Direct. A todo esto, Cassandra es una base de datos con miles de líneas de código en desarrollo durante más de 10 años. Esto es, modificar el código con la intención de no perder todas las prestaciones es de un trabajo inconmensurable. Así pues, esta implementación es mucho más sencilla, dejando de lado algunos servicios, en que las principales prestaciones de la base de datos han sido actualizadas.

Ahora, las tablas son guardadas en el dispositivo DAX directamente. La estructura y disposición de estas tablas forman un árbol, en que cada nodo es una tabla. Es decir, para acceder a una tabla, sea lectura o escritura, hay que recorrer un árbol. Por un lado, en la teoría resulta más lento que cuando se escriben memtables pero más rápido cuando estas son escritas en una SSD o HDD.

4.2.4 Experimentos

Para esta versión de Cassandra se han considerado cuatro experimentos: una aplicación exhaustiva de sólo lecturas, sólo escrituras, mixto de lecturas y escrituras y, por último, reinicialización de Cassandra. Una forma sencilla de obtener resultados sin necesidad de aplicaciones de terceros es utilizar la herramienta cassandra-stress[5], producida por los propios desarrolladores de Apache. Los tres primeros experimentos se van a llevar a cabo mediante este software. El último, volver a iniciar Cassandra, se realizará añadiendo líneas de

código en Cassandra para calcular el tiempo transcurrido desde su inicio hasta estar preparado a escuchar los clientes, habiendo cargado todas las tablas previamente guardadas en disco.

Primero, se realizarán las escrituras. Cassandra-stress permite configurar opciones y parámetros para ajustar los tests como queramos. Se van a realizar 10 millones de escrituras y, automáticamente, se crearán gráficos que representen la ejecución de estas operaciones. Los gráficos mostrados en este documento son los generados por el mismo cassandra-stress.

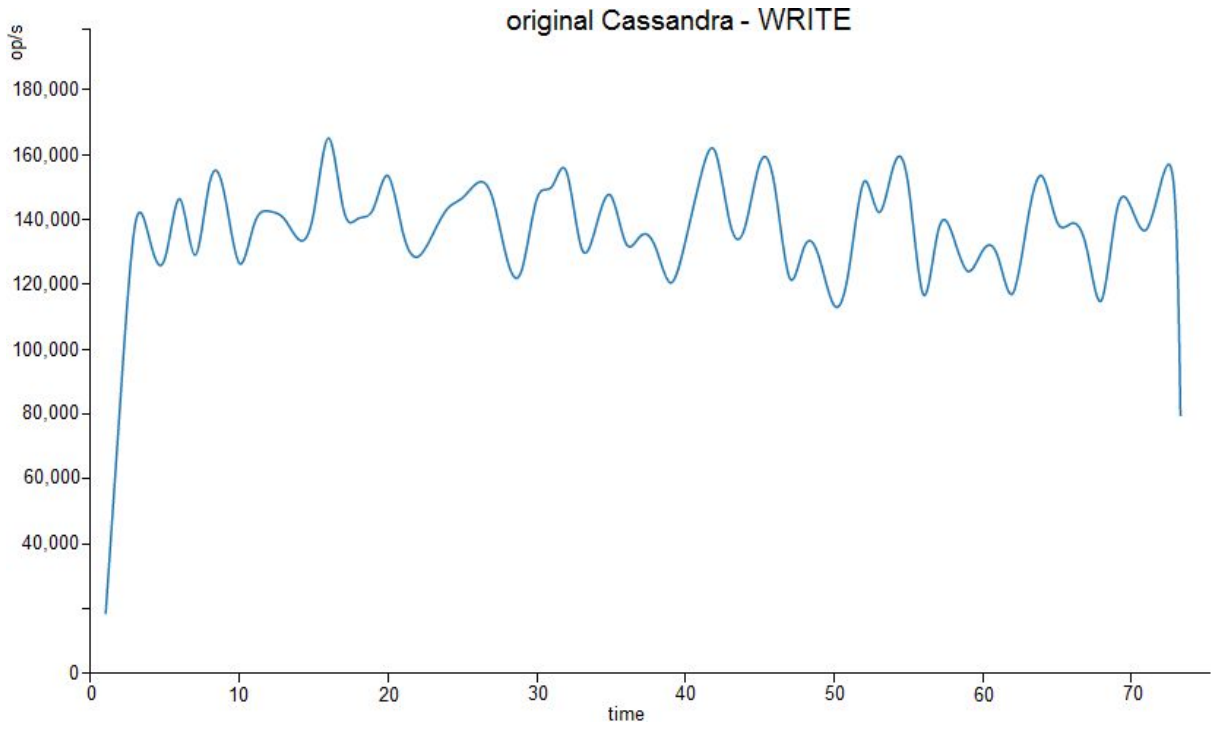

Fig. 9 Thoughput de escrituras del Cassandra original

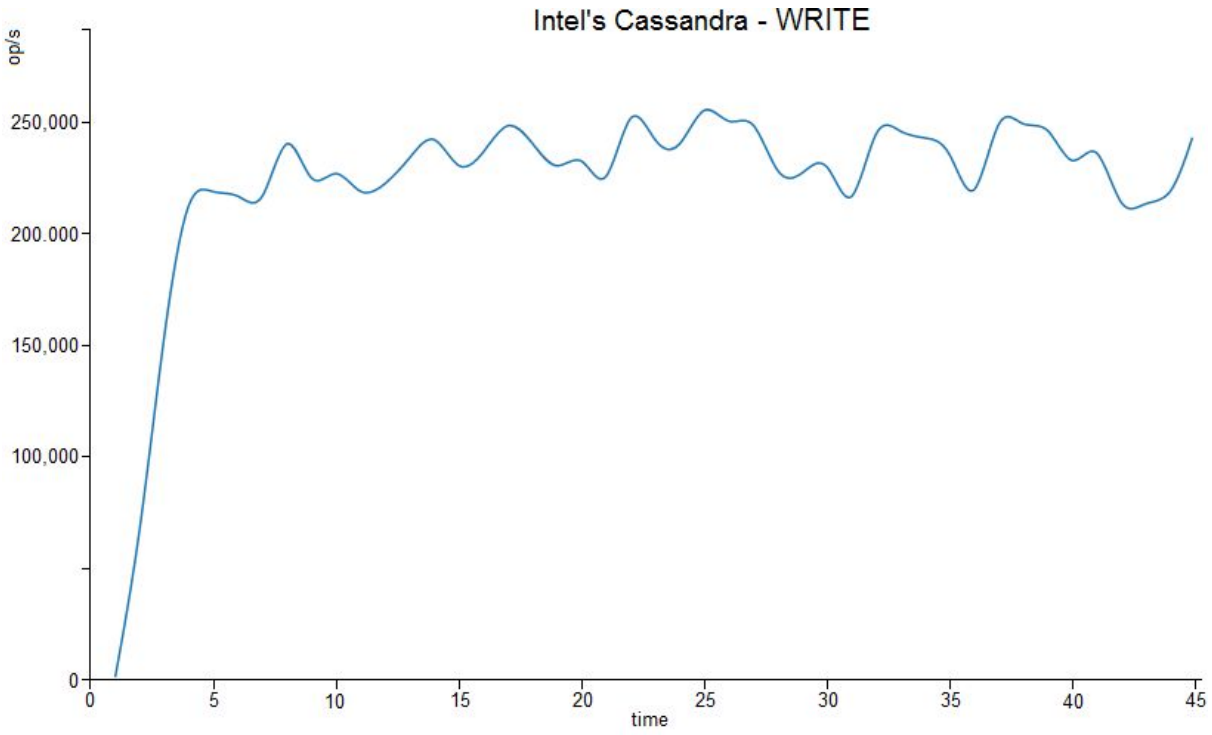

Fig. 10 Thoughput de escrituras del Cassandra persistente

	op/s	tiempo (s)	
Original version	136.432	73s	**ganancia**
Intel version	222.905	44s	65%

Fig. 11 Tabla de la comparativa (escrituras) entre el Cassandra original y el persistente

Vemos que el Cassandra persistente tiene mejor rendimiento. En general estos resultados han sido satisfactorios y se puede concluir que mejora las escrituras.

Seguidamente, se realizará la misma metodología para las lecturas. Las lecturas son sobre los datos escritos anteriormente, haciendo una lectura secuencial sobre estos.

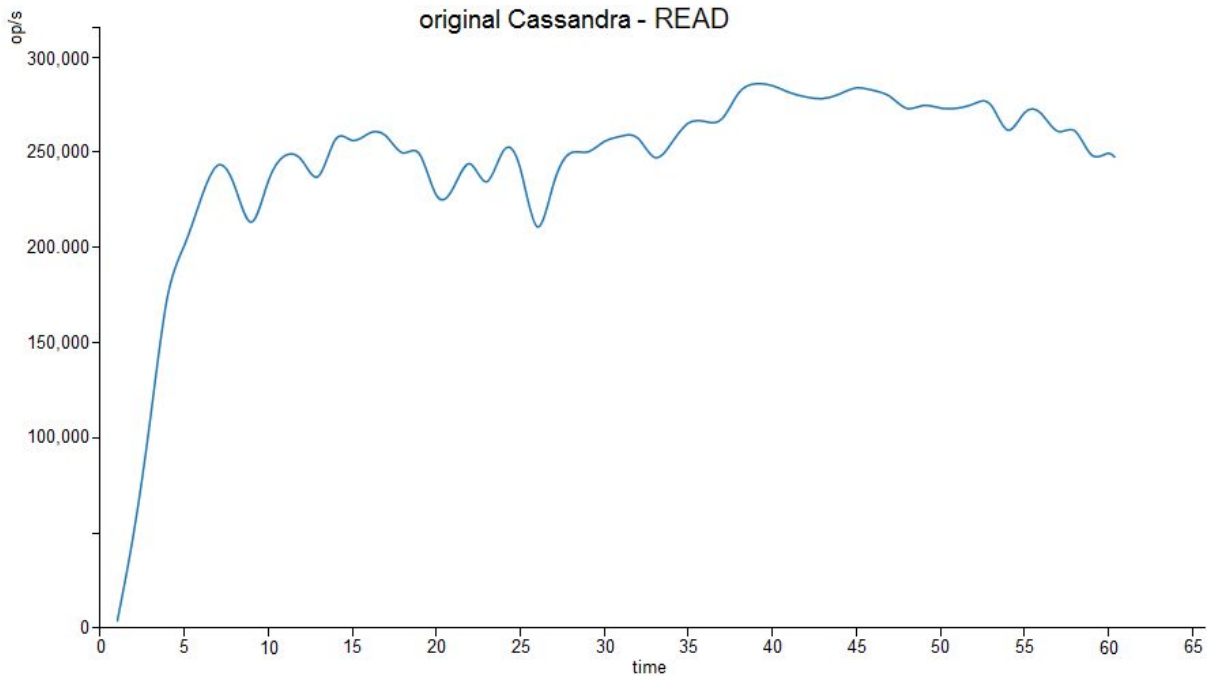

Fig. 12 Thoughput de lecturas del Cassandra original

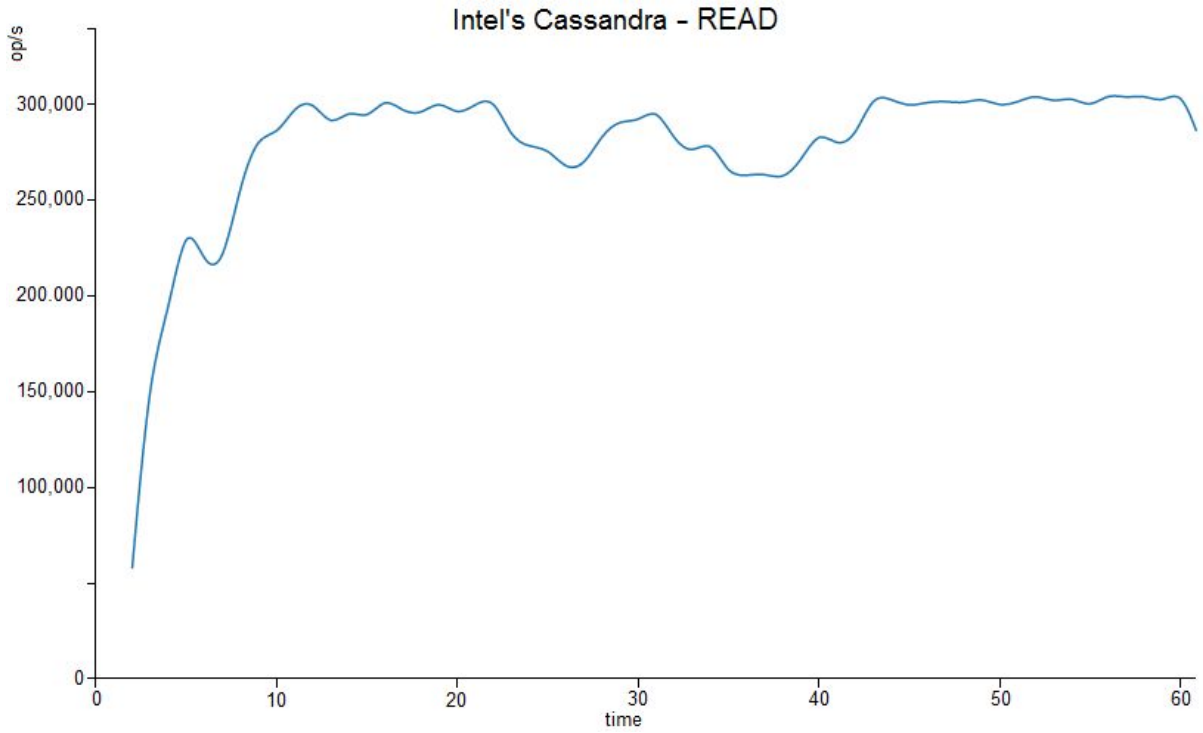

Fig. 13 Thoughput de lecturas del Cassandra persistente

	op/s	tiempo (s)	
Original version	244.852	36s	**ganancia**
Intel version	274.269	41s	12%

Fig. 14 Tabla de la comparativa (lecturas) entre el Cassandra original y el persistente

Se puede observar que en el Cassandra persistente sigue habiendo mejora. Sin embargo, la ganancia no ha sido tan grande como en las escrituras. El principal motivo es porque el Cassandra original se beneficia de la buffer cache del sistema operativo durante las lecturas secuenciales. A pesar de esto, los resultados siguen siendo satisfactorios.

Por último, la comanda mixed permite intercalar lecturas y escrituras:

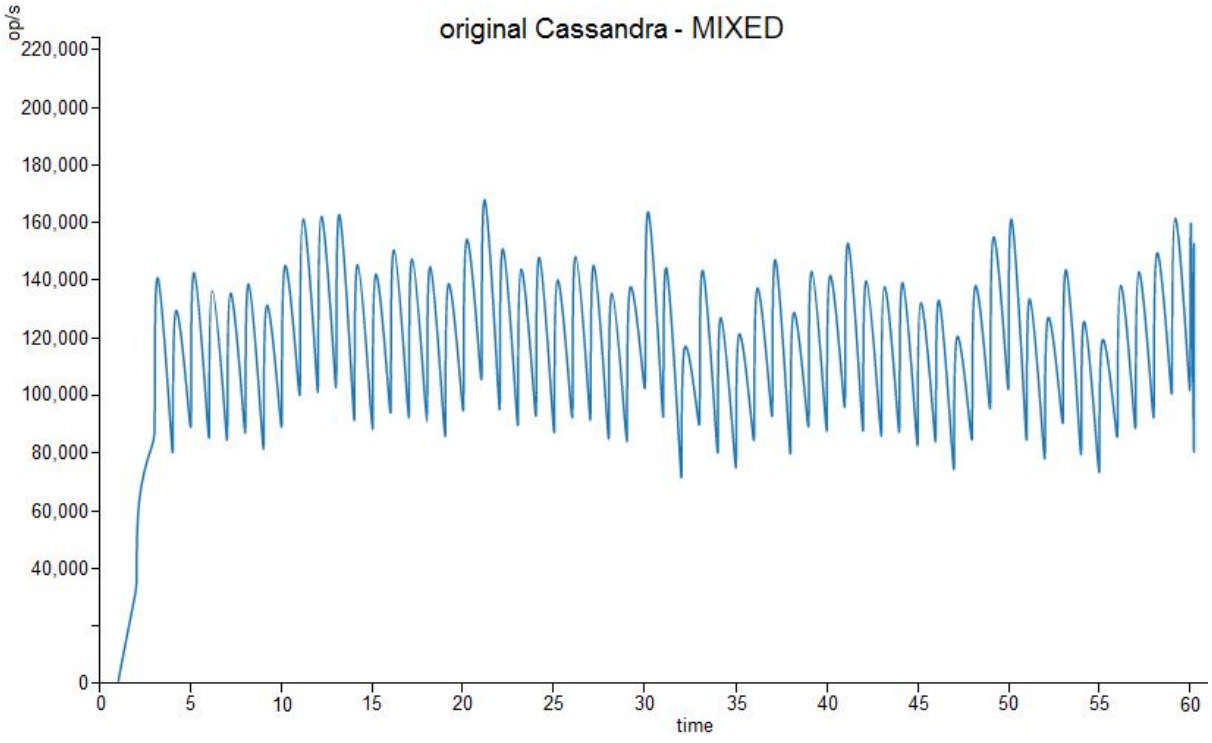

Fig. 15 Thoughput de lecturas y escrituras del Cassandra original

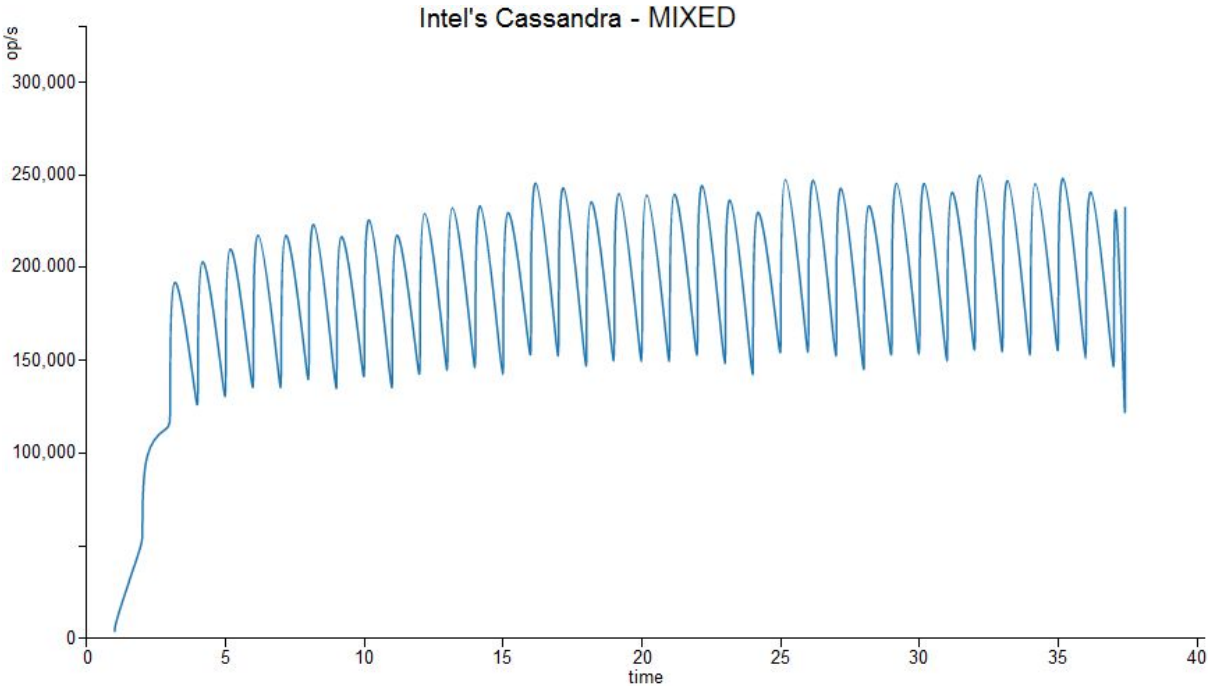

Fig. 16 Thoughput de lecturas y escrituras del Cassandra persistente

	op/s	tiempo (s)	
Original version	165.984	60s	**ganancia**
Intel version	267.253	37s	61%

Fig. 17 Tabla de la comparativa (mixed) entre el Cassandra original y el persistente

Teniendo en cuenta los anteriores resultados de las escrituras y lecturas, cabía de esperar este resultado. Efectivamente, hay ganancia. Destacar que, esta vez, la mejora ha sido equivalente tanto en lecturas como en escrituras.

Como parte final de los experimentos, se va a poner a prueba reinicializar Cassandra. Los datos que se van a cargar serán los producidos por las escrituras anteriores.

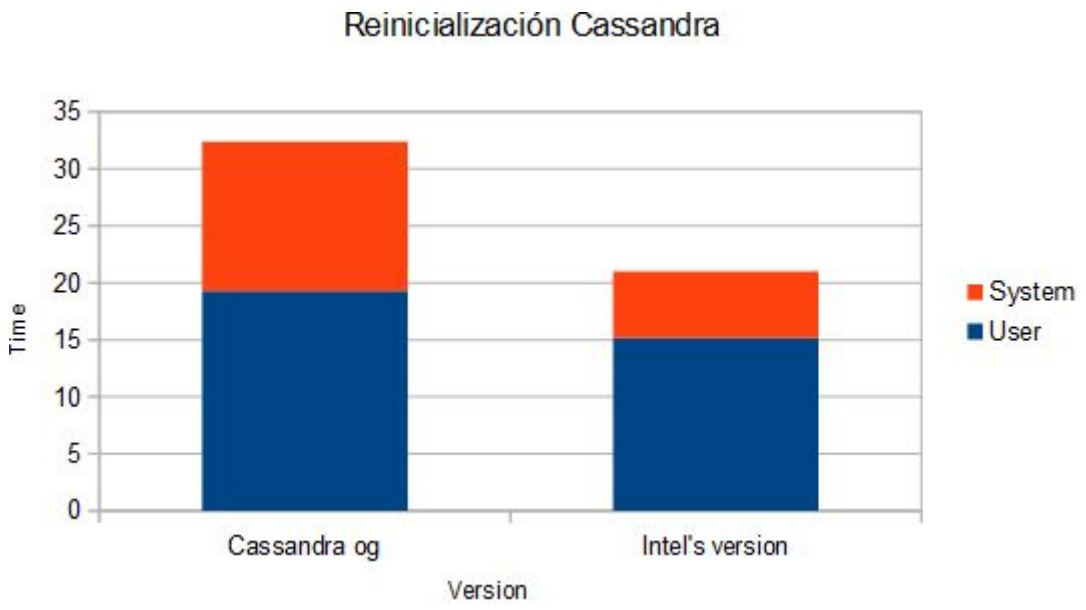

Fig. 18 Tiempos de reinicialización de las distintas versiones de Cassandra

	tiempo (s)	
Original version	32,39s	**ganancia**
Intel version	20,99s	54%

Fig. 19 Tabla de la comparativa (reinicialización) entre el Cassandra original y el persistente

Se puede observar como Cassandra persistente también es mejor. En general, los resultados han sido buenos a pesar de la simplicidad de los tests. Como trabajo en el futuro, se podrían hacer tests más exhaustivos y meticulosos para poder detallar el análisis, pero, para resolver los problemas planteados en un inicio, estos resultados han sido muy satisfactorios.

4.2.5 Conclusiones

Los resultados sacados de los experimentos son los que cabrían esperar en base a la teoría planteada. Hay cierta mejora en lecturas y escrituras, pero es verdad que no mucha. Aunque el Optane de Intel sea más rápido que el almacenamiento tradicional, no hay que ignorar el hecho que la implementación ignora muchas optimizaciones del Cassandra original. El nuevo código es muy simple, por lo que invertir más tiempo en su desarrollo podría explotar lo bueno de las estructuras originales de Cassandra y la persistencia de memoria de los DAX.

Por otro lado, donde también se puede ver el impacto del Optane es al reinicializar Cassandra. Cuando se inicia y detecta que tiene que cargar tablas del disco, el Cassandra persistente es mucho más rápido. Con que esta fase requiere de mucha actividad en el disco, cargar las tablas ubicadas en los DAX es más eficiente simplemente porque es más rápido que el HDD instalado en Apache Pass.

4.3 Cassandra persistente y Apache Arrow

El segundo problema a solucionar de este proyecto es adaptar el Cassandra persistente con el formato columnar de los datos que ofrece Apache Arrow. Primero se justificará el porqué de la utilidad del formato columnar y después el motivo del uso del software Apache Arrow. Por ello, hay que entender el almacenamiento estándar de las bases de datos convencionales y cómo afecta el formato columnar en ello. Por otro lado también se expone Apache Arrow y por qué es tan útil con los procesadores de hoy en día. Entonces, se modificará el Cassandra persistente como es debido. Finalmente, se someterá a unos tests para poder analizar los resultados y sacar conclusiones.

4.3.1 Introducción

En este apartado se va a introducir la implementación de Apache Arrow en el Cassandra persistente. Por defecto, las bases de datos escriben los datos en formato lineal. Apache Arrow nos proporciona una API cross-language para escribir cómodamente los datos en formato columnar (Figura 20). Este formato, junto con otras propiedades de Apache Arrow, debería habilitar una lectura de datos más rápida dependiendo de la transacción dada.

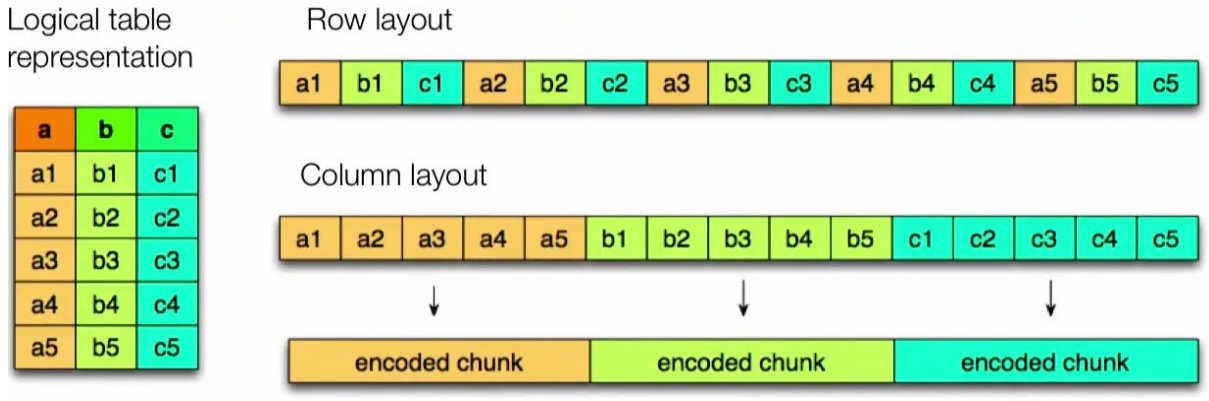

Fig. 20 Disposición de los mismos datos en formato lineal (arriba) y columnar (abajo)

El diseño de memoria columnar evita a las aplicaciones evitar operaciones I/O innecesarias y acelerar el rendimiento del procesamiento de datos en CPUs y GPUs modernas. En la figura 21 se puede ver como, dependiendo de los datos que pida la

transacción, la disposición de los datos en formato columnar en memoria nos permite lecturas más cómodas.

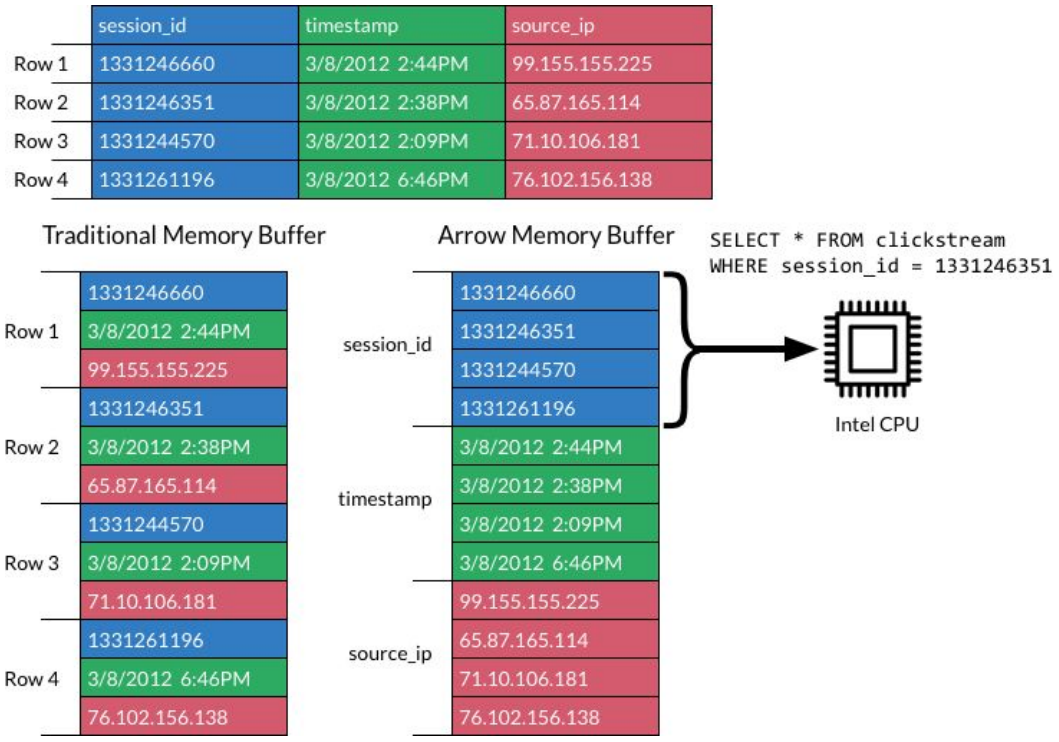

Fig. 21 Transacción de lectura que utiliza los datos de los buffers de Arrow

Apache Arrow permite a las aplicaciones aprovechar las últimas operaciones SIMD (Single instruction, multiple data) incluidas en los procesadores modernos, para la optimización de procesamiento de datos vectoriales. Este diseño columnar habilita que la localidad de datos permita un mejor rendimiento en CPUs de hoy día. También dispone de otras ventajas, pero no son relevantes para este trabajo[1].

A continuación se expone la implementación y se explicará cómo adherir Arrow en Cassandra y en el Optane. Cabe destacar que este código está orientado a un estudio de su rendimiento y no a una implementación real. Aún así, su diseño resultará intuitivo para el usuario y también facilitará su uso.

4.3.2 Diseño e implementación

En esta implementación de Apache Arrow se han tenido en cuenta varios criterios. Estos son: coherente con el diseño y estructuras del Cassandra persistente, poder explotar los beneficios de Arrow y sencillo para el programador y el usuario. Esta parte no está dirigida para proporcionar un servicio, se ha hecho desde un punto de vista de investigación y poner a prueba Arrow en una base de datos. Dicho lo anterior, primero se expondrá el diseño tomado y luego se justificará por qué cumple los criterios anteriores.

Para mantener la estructura de Cassandra, lo que se va a hacer será añadir en vez de modificar. Por cada tabla creada en Cassandra, se construirán otras que mantengan los mismos datos pero en formato Arrow. A partir de ahora, vamos a llamar estas tablas como tablas de Arrow. De esta forma, dependiendo de la transacción, podremos escoger las tablas de Cassandra o las nuevas tablas de Arrow. Como se ha expuesto en el apartado anterior, para obtener las máximas capacidades de Arrow, se utilizarán las transacciones que requieran de todos los datos de una o más columnas de una tabla. De esta forma, se aprovechará la disposición de los datos en memoria y las propiedades de las SIMD que Apache Arrow quiere explotar.

Entonces, cómo van a ser estas tablas. Para poder disponer los datos contiguos de las columnas en memoria, primero habrá que disponer de la información de varias líneas. Esto quiere decir que primero tendremos que acumular varias filas y, llegado a un límite, se escribirán todos estos datos en formato columnar. En la Figura 21 se puede observar el caso con cuatro filas.

Una forma sencilla de implementarlo, sin necesidad de modificar las estructuras de Cassandra, es creando una tabla con funcionalidad de buffer. Cada fila de esta tabla contendrá el *token_id*, el *timestamp* y la información de las columnas de la fila escrita en la tabla de Cassandra.

- ***Token_id***: es la *primary key* hasheada. Es un elemento que utiliza Cassandra para poder agilizar la ubicación de los datos y sus lecturas con indexación. En este trabajo, la gestión de este atributo es muy simple y servirá si se continúa el desarrollo del proyecto en el futuro.
- ***Timestamp***: momento en que se ha hecho la escritura. Servirá para ordenar todas las escrituras del mismo *token_id*. Como en el token_id, su uso será simple.
- ***Partition update* (serialized_pu)**: información de la fila. Aquí nos interesa los datos de las columnas y se va a utilizar cuando por fin se escriban estos datos en formato columnar.

En la siguiente figura se muestra la parte de código que construye esta tabla.

```
//buffer table
String tableBufferStatement = "CREATE TABLE IF NOT EXISTS " + arrowKeyspace + '.' + bufferTable +
                              " (token_id bigint, " +
                              "ts timestamp, " +
                              "serialized_pu blob, " +
                              "PRIMARY KEY (token_id, ts) );";
QueryProcessor.executeOnceInternal(tableBufferStatement);
```

Fig. 22 Código de la construcción de la tabla buffer

Entonces, a partir de un número de filas escrito, se pretende escribir los datos en formato columnar. Una solución a esta gestión es otra nueva tabla. Primero se van a nombrar el contenido de las columnas y luego su propósito.

- ***Token_id***: *primary key* hasheada.
- ***Timestamp***: momento en que se ha hecho la última escritura.
- **Dirección Arrow**: ubicación del fichero Arrow en el dispositivo DAX.
- **Tamaño Arrow**: tamaño del fichero Arrow.

Apache Pass tiene dos DAX (Figura 23). Uno es utilizado por Cassandra, donde guarda todas las tablas, etc. El otro queda libre, el cual sea ha aprovechado para guardar los datos en Arrow. Si queremos persistencia en en estos datos, la API de Apache Arrow nos permite escribir ficheros en formato columnar. Aprovechando la rapidez del Optane de Intel, estos ficheros son serializados y guardados en este Optane, en vez de en los SSD. Aquí es donde entra en juego los atributos *arrow_addr* y *arrow_size* de esta tabla (Figura 24). En esta tabla no se guardan los datos en sí, sino la referencia en donde estos están ubicados (*arrow_addr*). El *arrow_size* nos permite saber el tamaño de los ficheros, necesario para cuando los queramos leer.

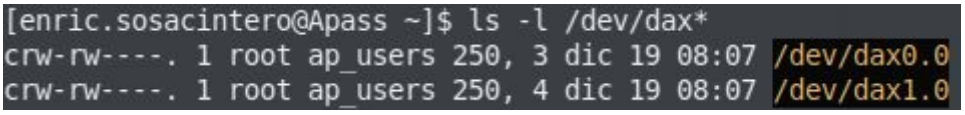

Fig. 23 Dispositivos DAX disponibles en Apache Pass

```
//arrow table
String tableArrowStatement = "CREATE TABLE IF NOT EXISTS " + arrowKeyspace + '.' + arrowTable +
                             " (token_id bigint, " +
                             "ts timestamp, " +
                             "arrow_addr bigint, " +
                             "arrow_size int, " +
                             "PRIMARY KEY (token_id , ts) );";
QueryProcessor.executeOnceInternal(tableArrowStatement);
```

Fig. 24 Código de la construcción de la tabla buffer

En conclusión, por cada tabla creada en Cassandra (por ejemplo, con un *CREATE TABLE* de un usuario), van a ver dos más: la primera, que actúa como buffer el cual se vaciará llegado un threshold, después de escribir los datos guardados en formato columnar en un DAX; y la segunda, que apunta donde están guardados los ficheros Arrow. En la siguiente figura se visualiza el esqueleto de este diseño.

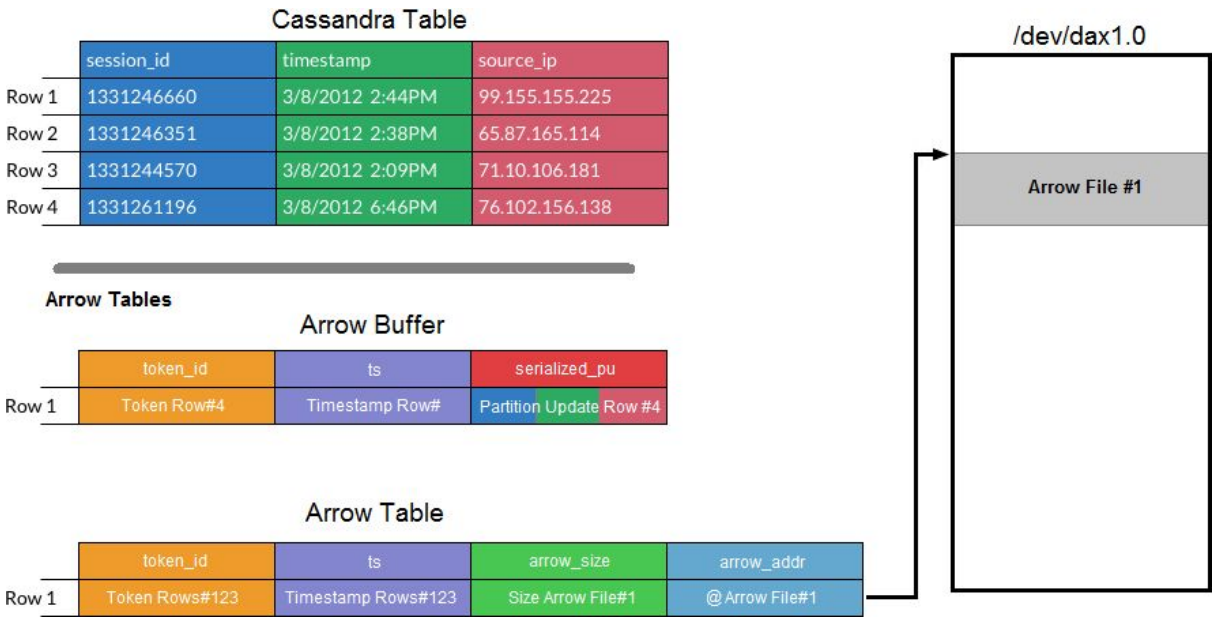

Fig. 25 Diseño esquemático del Cassandra con Apache Arrow

Como último remarque, sólo se han tenido en cuenta transacciones que escriben nuevas filas en las tablas. Esto es, esta implementación sólo soporta transacciones *INSERT INTO* pero no gestiona actualizaciones (*UPDATE*) o eliminaciones (*DELETE*). De esta forma se simplifica mucho el diseño, se evita una gestión propia de las bases de datos -competencia que no se profundiza en este trabajo- y se centra en la explotación de las propiedades de la librería Apache Arrow.

4.3.3 Experimentos

Como se ha expuesto anteriormente, la implementación de Arrow resulta delicada. Esto es, ahora no nos podemos valer de cassandra-stress, sino de unos tests más específicos diseñados por nosotros. Una forma cómoda de hacerlo es mediante ficheros con instrucciones CQL (Cassandra Query Language), lo cual nos permite definir las tablas y los tipos de datos que contienen. Además, también servirá para realizar la cantidad de escrituras que queramos, con los datos que queramos.

Con Apache Arrow, se espera que las lecturas que seleccionen todos los datos de una/s columna/s sean más rápidas. Por lo tanto, los experimentos tienen que cumplir estas necesidades. Así pues, los primeros tests son la lectura de una tabla con un número determinado de filas. La comparativa va a ser entre una tabla de Cassandra creada con un *CREATE TABLE* y las nuevas tablas con Arrow. Después de realizar la transacción de lectura de estas tablas, se va a calcular una suma de todos los enteros de una columna.

Antes de especificar los experimentos, se aclara que durante la lectura de las tablas de Arrow no se ha tenido en cuenta el buffer. Es decir, en algunos casos varios datos pueden encontrarse en el buffer durante una lectura, pero en los tests se ha pasado por alto estas

situaciones. En los tests se asegurara que no haya datos en el buffer, pero se pasa por alto su consulta. De momento el estado actual del proyecto es un prototipo, así que esta prestación todavía está por desarrollar. Por último, decir que sólo se ha utilizado un cluster.

Para que la comparativa sea lo más justa posible, se han hecho varias tomas con cada una de las tablas. Cuando se hace un *select* de una tabla con muchas filas, Cassandra prefiere que se haga con paginación (*paging*[13]). Por otro lado, tenemos los datos en formato columnar. Un parámetro a tener en cuenta a la hora de escribir los datos en este formato es el tamaño de los bloques de datos escritos. Este parámetro lo llamaremos *batchSize*. Teniendo esto en cuenta, estos son los resultados sacados:

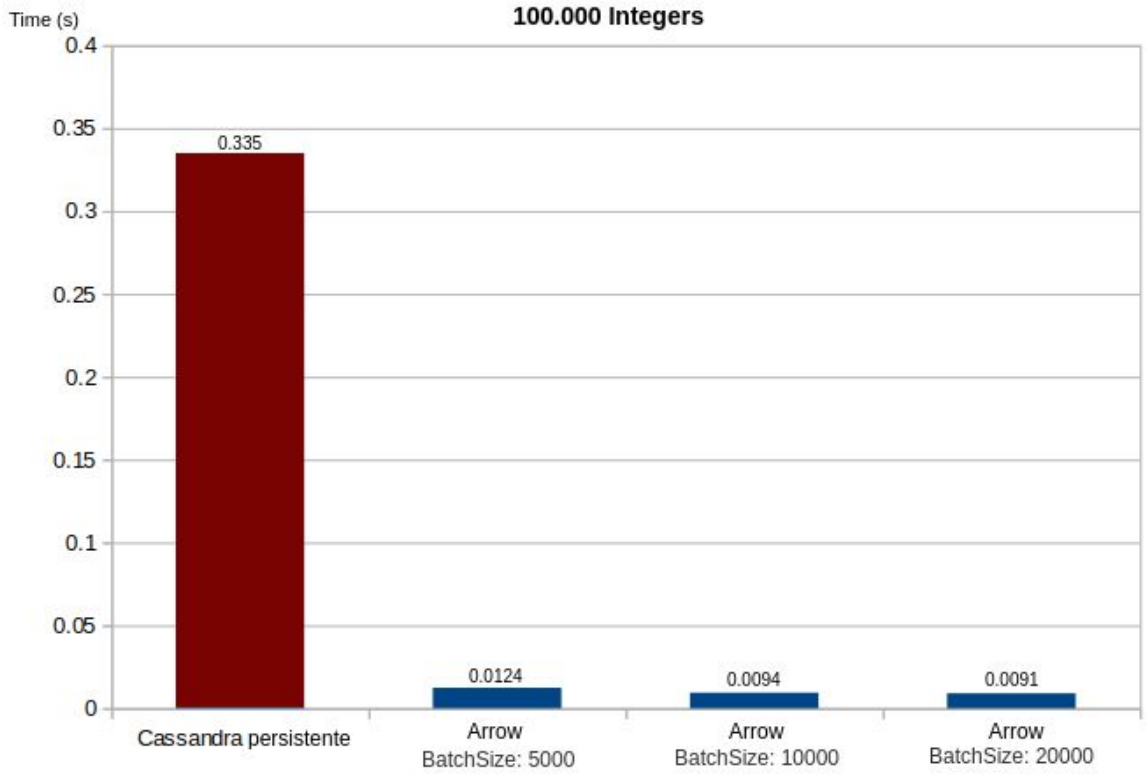

Fig. 26 Lectura de 100.000 enteros, comparativa entre Cassandra persistente y Arrow

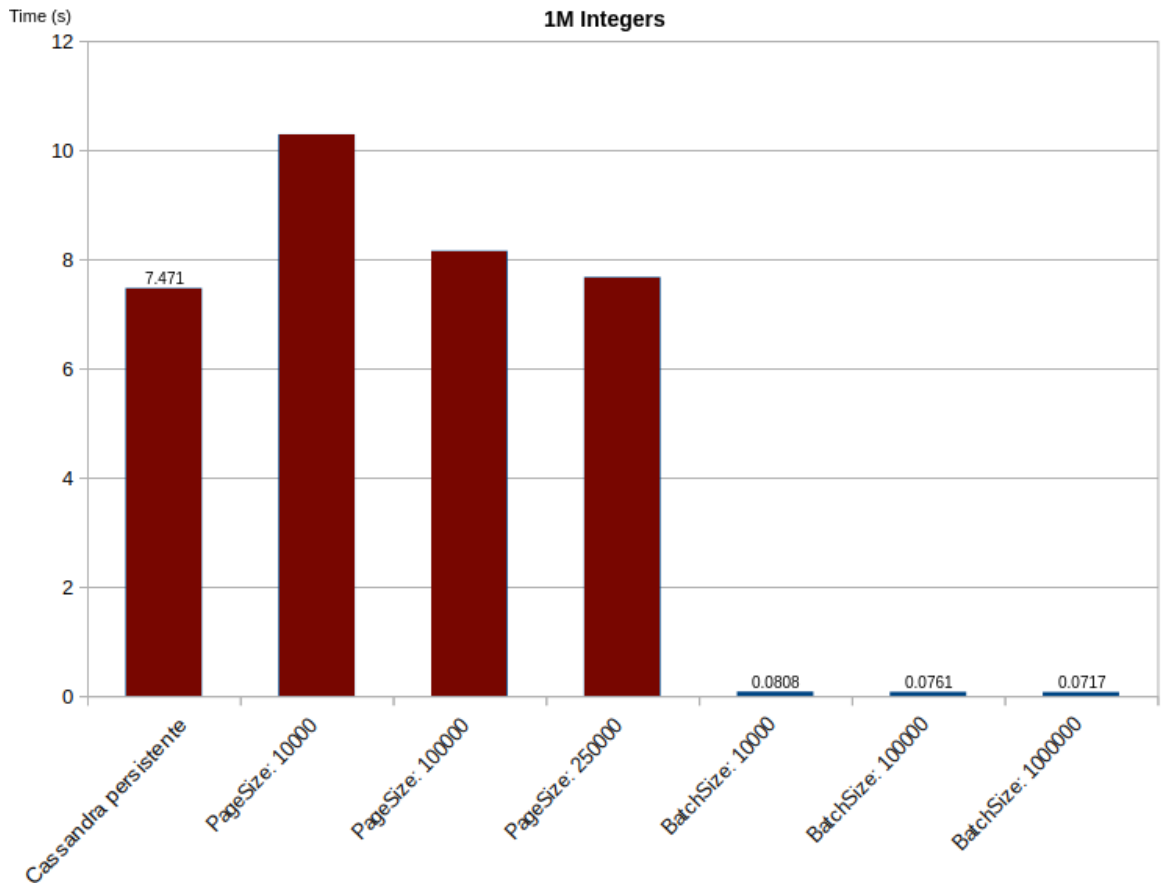

Fig. 27 Lectura de 1.000.000 enteros, comparativa entre Cassandra persistente y Arrow

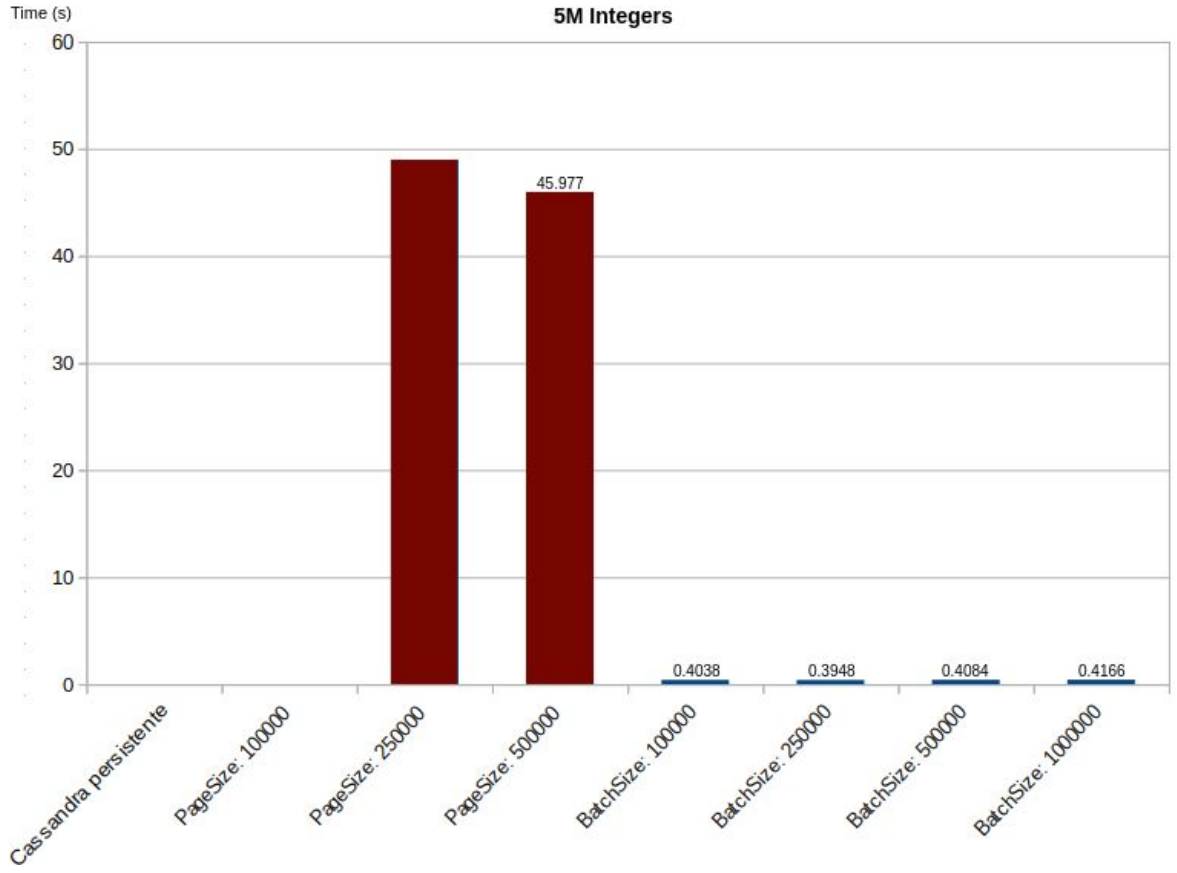

Fig. 28 Lectura de 5.000.000 enteros, comparativa entre Cassandra persistente y Arrow.

En esta última figura (Fig. 28), los dos primeros casos (el Cassandra persistente sin paginación y con paginación de tamaño 100.000) dieron *timeOut* durante la transacción de lectura.

# Filas leídas	Mejor tiempo Cassandra (s)	Mejor tiempo Arrow (s)	ganancia
100.000	0,33497	0,00907	36,96
1.000.000	7,47082	0,07167	104,24
5.000.000	45,97647	0,39578	116,17

Fig. 29 Tabla de la comparativa (lecturas) entre el Cassandra persistente y el Arrow

En efecto, estas lecturas mejoran con las nuevas tablas. El speedup resultante es tan alto porque, en lecturas de muchos datos, Cassandra es muy lento. Esto, sumado con las propiedades que explota Apache Arrow, deriva a una ganancia enorme.

Por último, toca calcular el overhead añadido por las nuevas tablas. Como es obvio, haber agregado esta prestación tiene un coste. Lo que se ha hecho en estas últimas pruebas es comparar el tiempo de las escrituras entre el Cassandra persistente y el Cassandra persistente con Arrow. Cabe destacar que en ambos casos se replican los datos de Cassandra una vez. Por lo tanto en estos tests, utilizando Arrow se hace réplica dos veces (formato de Cassandra y formato columnar de Arrow). En un caso real, utilizando Arrow, habría una sola réplica con los datos en formato columnar.

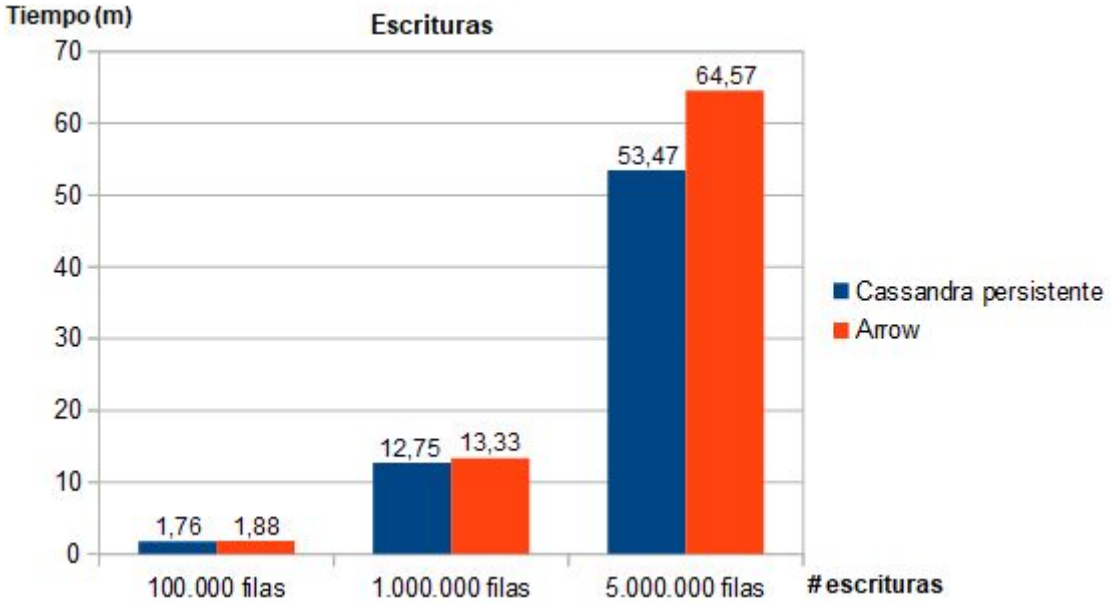

Fig. 30 Escrituras, comparativa entre Cassandra persistente y Arrow

# Filas escritas	Tiempo Cassandra (m)	Tiempo Arrow (m)	ganancia
100.000	1,76	1,88	-6,38%
1.000.000	12,75	13,33	-4.35%
5.000.000	53,47	64,57	-17,19%

Fig. 31 Tabla de la comparativa (lecturas) entre el Cassandra persistente y el Arrow

Los resultados son coherentes y la versión con Arrow hace las escrituras más lentas. Sin embargo, en comparación con la ganancia de algunas lecturas sí que vale la pena.

4.3.4 Conclusiones

La ganancia conseguida con las lecturas es enorme. Cabe destacar que el formato columnar sólo servirá para algunas lecturas, pero la mejora vale la pena. Como se ha dicho antes, durante la lectura de las tablas de Arrow no se consulta la tabla buffer. Sin embargo, con las lecturas con la tabla de Cassandra se ha intentado también ser justo. Por ello, se ha utilizado la metodología de paginación, lo cual es muy común dentro del uso de Cassandra.

Aunque tenga un obvio overhead y su uso sea ocasional, la ganancia es demasiado grande como para ignorarla. En un trabajo futuro, habrá que hacer análisis más extensos con diversos tipos de datos, e intentar ver cómo mejorar la actual versión.

5 Conclusiones del proyecto

Durante el capítulo anterior se ha desarrollado la implementación del proyecto. Asimismo, también se ha hecho experimentos para probar el diseño y un análisis de los resultados sacados. Sin embargo, falta por concluir el proyecto exponiendo si se ha resuelto el problema planteado al principio y el trabajo futuro a raíz del trabajo.

5.1 Resolución del problema

En el apartado **1.2 Formulación del problema** se expusieron los problemas a resolver de este proyecto. Resumiendo, estos son:

1. Comparar, con esperanza de mejorar, el rendimiento de Cassandra con la tecnología de Intel.

Esta investigación se ha desarrollado durante el subapartado **4.2. Cassandra persistente**. En la base de datos se han tomado varias medidas para determinar resultados más concretos. De esta forma, se pretende reforzar la veracidad de las comparaciones realizadas. Por lo tanto, este requisito se ha cumplido satisfactoriamente.

2. Agilizar algunas lecturas con las posibilidades que nos ofrece Apache Arrow.

Esta investigación se ha desarrollado durante el subapartado **4.3. Cassandra persistente y Apache Arrow**. A pesar del permanente overhead en las escrituras, la ganancia conseguida en algunas lecturas resulta tan grande que se hace difícil pasar por alto los resultados. El problema planteado ha sido poco concreto, pero era de esperar que, al añadir código de más, también haría ralentizar algunas partes de Cassandra. Así pues, este problema también se a resuelto satisfactoriamente.

Además de los problemas expuestos, también se pretendió que el proyecto cumpliese unos requisitos con tal de agilizar el progreso. Estas condiciones están definidas en el subapartado **2.1.2. Requisitos del proyecto**. En general, especifican la forma en que se debería implementar el código. Aunque los detalles se han pasado por alto en la memoria del proyecto, sí que se ha determinado su diseño y se ha justificado su estructura.

En conclusión, el colofón de la investigación ha resultado satisfactorio. Durante el proyecto se ha pasado por alto que la implementación no apoye todos los casos o que no se haya profundizado en algún tema. Así pues, en el siguiente apartado se expondrá las posibles mejoras o caminos que podría tomar este proyecto.

5.2 Trabajo futuro

Los resultados han sido satisfactorios, pero no por ello la investigación ha terminado. Aún hay margen de mejora, y se piensa que puede valer la pena continuar, en base a los resultados extraídos. Por ejemplo, la profundidad de los experimentos no ha sido muy alta; el contexto de la ejecución ha sido siempre el mismo y el análisis sacado tampoco detalla el comportamiento del programa. Además, la implementación de Intel también ha sacrificado algunas prestaciones del Cassandra original, lo cual se debería de notificar cuales han sido y la gravedad de cada una.

Por otro lado, dejando de lado las posibles mejoras, el proyecto puede resultar de gran utilidad dentro del grupo de trabajo en Computer Science del BSC. De esta forma, este trabajo puede formar parte del desarrollo de aplicaciones o investigaciones de más alto nivel.

6 Presupuesto del proyecto

Aquí es donde se va a especificar la gestión económica del proyecto. No menos importante, también hay que tener en cuenta los costes indirectos, como gastos de la oficina u otros sujetos externos. Finalmente, se calcula el presupuesto final y éste se va a desglosar según las tareas especificadas en la planificación.

En cada apartado se detallarán los costes calculados, amortizaciones, contingencias e imprevistos. Si es el caso, también se describirán los mecanismos para controlar las gestiones de desviaciones que puedan aparecer en relación con el presupuesto, definiendo también indicadores numéricos de cálculo que faciliten el control.

6.1 Costes humanos

El coste de los recursos humanos se puede detallar según las ocupaciones necesarias y la cantidad de horas dedicadas de cada una de las ellas. En base a esto, se hará una búsqueda del salario de estas ocupaciones.

Los sujetos implicados son: el investigador, como ingeniero de investigación junior; la directora y el profesor de GEP, como gestores del proyecto. También, del departamento de Computer Science del BSC: desarrollador de software junior y senior. Una vez identificados los roles, estimamos su salario gracias a la página web: https://www.payscale.com/research/ES/Country=Spain/Salary.

Rol	Coste por hora (€/h)
Ingeniero de Investigación junior	10,00
Gestor de proyectos	28,00
Desarrollador de software junior	21,00
Desarrollador de software senior	26,00

Fig. 32 Tabla de coste por hora de cada rol

A continuación, calculamos las horas dedicadas por cada rol según la tarea:

Tarea	Horas (h)	Horas por rol (h)			
		Ing. junior	Gestor	Des. junior	Des. senior
Gestión del proyecto	**75**	10	65	-	-
Análisis y diseño	**60**	45	-	5	10
Cassandra persistente	**120**	105	-	-	15
Apache Arrow	**100**	85	-	-	15
Evaluación de rendimiento	**40**	35	-	5	-
Hito final del proyecto	**75**	65	10	-	-
Total	**470**	345	75	10	40

Fig. 33 Tabla de horas de cada rol

Finalmente, calculamos el presupuesto total:

Rol	Horas (h)	Coste (€)
Ingeniero de Investigación junior	375	3.750,00
Gestor de proyectos	75	2.100,00
Desarrollador de software junior	10	210,00
Desarrollador de software senior	40	1.040,00
Total	470	7.100,00

Fig. 34 Tabla del total de horas y coste de los recursos humanos

Hay que tener en cuenta que los imprevistos afectan directamente a la cantidad de horas total. Especialmente, en las fases de implementación (Cassandra persistente y Apache Arrow) es donde puede haber más contratiempos. Por lo tanto, en los recursos humanos es donde puede haber más desviación económica, afectando principalmente al investigador junior y al desarrollador senior.

6.2 Costes hardware

Se especifican los precios de los recursos hardware. También se tiene en cuenta la pantalla, teclado y ratón:

Hardware	Coste (€)	Vida útil (años)	Amortización (€)
Ordenador Portátil Dell	1.219,00	4	304,75
Pantalla Dell	225,00	4	56,25
Teclado Logitech	20,00	4	5,00
Ratón Logitech	15,00	4	3,75
Máquina Apache Pass	21.500,00	2	-
Total	22.776,00	-	369.75

Fig. 35 Tabla de costes hardware

La variabilidad en el coste de los recursos hardware debería ser bastante bajo. El mayor problema es si se estropease el portátil, pero se puede apañar utilizando otro de más barato. Sin embargo, hay que tenerlo en cuenta.

6.3 Costes software

Convenientemente, todos los recursos software utilizados son gratuitos. Cabe destacar IntelliJ IDEA Ultimate, el cual no es de pago si se verifica que eres estudiante[14].

Software	Coste (€)	Vida útil (años)	Amortización (€)
Apache Cassandra	0.00	-	0.00
Apache Arrow	0.00	-	0.00
Google Drive	0.00	-	0.00
LibreOffice	0.00	-	0.00
IntelliJ IDEA Ultimate	0.00	-	0.00
Slack	0.00	-	0.00
Google Calendar	0.00	-	0.00
OverLeaf	0.00	-	0.00
Ubuntu 18.04	0.00	-	0.00
Atenea	0.00	-	0.00
Total	0.00	-	0.00

Fig. 36 Tabla de costes software

La única desviación posible es si se acaba la licencia de IntelliJ IDEA Ultimate antes que se acabe la fase de implementación y testing, pero no es el caso. Aún así, se tendrá en cuenta que se puede utilizar un software de pago que no se esperaba en un inicio.

6.4 Costes indirectos

Principalmente incunve a los gastos en la oficina, ubicada en el edificio C6 del Campus Nord:

Recurso	Coste (€)
Oficina	5.000,00
Mobiliario oficina	1.500,00
Prestaciones oficina	3.000,00
Total	9.500,00

Fig. 37 Tabla de gastos indirectos

Los números mostrados son muy orientativos, por lo que seguramente va a haber modificaciones y no se puede detallar un control de gestión de desviaciones.

6.5 Costes de gestión

Como todo proyecto es candidato a padecer desviaciones en la planificación y, consecuentemente, en el presupuesto. A pesar de que se hayan mencionado diversas técnicas y metodologías para evitar contratiempos, no implica que sea imposible encontrarse con adversidades.

El presupuesto relacionado con los recursos humanos es el más probable que se vea afectado. Las horas elucubradas en la sección **2.3. Planificación temporal** han sido aproximadas y no se puede saber exactamente su duración. La duración de las tareas puede verse alterada y esto implica tambień cambios en las demás tareas relacionadas. Principalmente, las tareas relacionadas con el entorno informático: implementación y tests. En estas secciones es donde fácilmente pueden surgir dificultades repentinas. Las horas del investigador y, en menor medida, las del desarrollador de software senior, van a ser las más inestables, por lo que el presupuesto de los imprevistos tiene que estar calculado acorde a estos salarios.

Sin embargo, los costes hardware no se verán muy alterados. Por un lado, su coste es exacto y su amortización no se ve tan afectada con la posible variabilidad de horas de trabajo. Por otro, sólo accidentes graves afectarían notablemente los recursos hardware, como virus letales o daños físicos. Las probabilidades que esto pase son prácticamente nulas.

El presupuesto software sólo puede incrementar si se utiliza uno de pago que no se había planeado. En principio, se han tenido en cuenta todos los recursos necesarios.

En cuanto a los costes indirectos, ya se ha especificado que son muy orientativos y no se pueden especificar mecanismos para controlar desviaciones.

A continuación, se detalla en una tabla el presupuesto del proyecto teniendo en cuenta la contingencia y los imprevistos:

Actividad	Coste (€)	Riesgo	Observaciones
Contingencia	2.545,46	-	Margen de seguridad, calculado como un porcentaje del valor total calculado en un 15%, en base, esencialmente, al presupuesto de los recursos humanos.
Imprevistos	-	-	Imprevistos que no salen directamente en el Gantt. Identifican los riesgos (Sección 3.4 Riesgos), los costes de sus alternativas y los niveles de probabilidad.
Virus / error fatal en el portátil (Coste: ~800€)	80,00	10%	La alternativa será utilizar otro ordenador portátil de menor calidad. En el coste se ve reflejado el importe de un nuevo ordenador de menor calidad.
Error crítico en Apache Pass (Coste: ~520€)	26,00	5%	La alternativa será virtualizar el Optane de Intel. En el coste se ve reflejado, de forma aproximada, las horas extras que tendrán que trabajar: investigador (~25h de análisis e implementación), el desarrollador de software senior (~5h de análisis y diseño) y el gestor de proyectos (~5h de guía del proyecto y gestión de la comunicación del equipo).
Total	2.651,46	-	-

Fig. 38 Tabla de gastos de contingencia e imprevistos

6.6 Presupuesto total

Una vez calculados todos los costes, es posible calcular el presupuesto final estimado de este proyecto:

Recurso	Coste
Recursos humanos	7.100,00
Recursos hardware	369,75
Recursos software	0,00
Costes indirectos	9.500,00
Contingencia e imprevistos	2.651,46
Total	19.621,21
Total (IVA +21%)	23.741,66

Fig. 39 Tabla con el presupuesto final calculado

6.7 Desviaciones y presupuesto final

A pesar de ciertos imprevistos y alguna desviación, no han tenido un impacto directo en el presupuesto. Si bien es cierto que alguna tarea ha costado más de lo esperado, el coste de tiempo no ha sido mucho mayor, además de haber tenido la posibilidad de hacer otras para no malgastar tanto tiempo atascado. Sumando el hecho de que ya hemos tenido en cuenta el coste de contingencia e imprevistos, el presupuesto total calculado en el apartado anterior se adecúa bastante al final.

7 Informe de sostenibilidad

En este capítulo se va a reflexionar sobre la sostenibilidad del proyecto y su impacto social. Este tema se suele pasar por alto pero su relevancia cada vez se hace notar más. El análisis va a tener en cuenta el impacto sobre el medio ambiente y su huella ecológica, el consumo de recursos materiales y humanos y el coste de dichos recursos y la influencia en las personas que han trabajado en este proyecto. En las siguientes secciones se van a desarrollar estas competencias según su dimensión económica, ambiental y social. Asimismo, estas dimensiones se van a clasificar según el estado del proyecto: Proyecto Puesto en Producción (PPP), vida útil y riesgos.

7.1 Proyecto Puesto en Producción (PPP)

Incluye la planificación, el desarrollo y la implementación del proyecto.

7.1.1 Dimensión Ambiental

La oficina y el material informático suman la mayoría del impacto ambiental en recursos materiales. Cuantificar el uso de las instalaciones de la oficina resulta difícil por el hecho de estar compartidas por otros muchos trabajadores. Además, no es posible el acceso a la facturación del consumo. Por otro lado, tenemos el material informático, el cual mayoritariamente se reduce a su consumo de energía eléctrica. Cuando más se consume es durante su uso, por lo que se ha intentado minimizar el consumo apagando y desactivando el material cuando éste no ha sido necesario.

Por último, y no menos importante, el personal involucrado también ha tenido relevancia en esta sección. De forma indirecta, el transporte a la oficina, el uso de prestaciones de ésta, etc. también han tenido su impacto ambiental.

7.1.2 Dimensión Económica

Los costes de recursos humanos y materiales durante la realización del proyecto han sido especificados y calculados en el apartado **6. Presupuesto del proyecto**. En la planificación temporal, detallada en **2.3. Planificación temporal**, también se ha visto cómo ha influenciado en el presupuesto del desarrollo del proyecto. Dado que la planificación inicial se ha ajustado bastante bien al desarrollo del proyecto, el coste calculado se ha ajustado al presupuesto final. Además, a pesar de algunas desviaciones, el coste final no ha superado al coste inicial por haberle añadido un porcentaje de imprevistos y contingencias.

También cabe destacar la metodología utilizada y la definición de posibles concurrencias entre tareas, que han agilizado el desarrollo del proyecto. Por lo que se refiere a la metodología, el proceso iterativo ha ayudado a acotar el impacto económico por la constante planificación que requería. Por último, gracias a algunas concurrencias entre tareas, algunos imprevistos no han perjudicado al progreso por tener la posibilidad de trabajar en otras labores.

7.1.3 Dimensión Social

A nivel ético y moral el proyecto no ha planteado ninguna reflexión. A causa de su inherente neutralidad en este ámbito no han surgido controversias colaterales. Igualmente, en **1.3. Agentes implicados** ya se han definido los roles implicados en el proyecto y cómo les afecta.

A nivel personal, haber podido trabajar con tecnología y software reciente con expectativas de futuro con el fin de investigar, resultará muy fructífero para mi formación. Además, ser el principal responsable de un proyecto de esta magnitud me ayudará a organizarme mejor y planificar proyectos adecuadamente, asimismo como tener en cuenta su impacto colateral en el contexto en el que se ha trabajado.

7.2 Vida Útil

Incluye el transcurso del proyecto una vez puesto en producción: empieza una vez implementado y acaba con su desmantelamiento.

7.2.1 Dimensión Ambiental

Gran parte del proyecto es software, por lo que de forma directa no consumirá recursos materiales. Respecto al hardware, el Optane de Intel, como todo dispositivo informático gastará energía eléctrica. Si los conocimientos de este proyecto se aplican en uno que ya ha salido del PPP, el consumo extra causado por la memoria de Intel fácilmente podría ser negligible respecto al consumo de potencia de cálculo de la aplicación.

Como se ha demostrado que este proyecto puede mejorar el rendimiento de una base de datos, esta mejora supone menor inversión en horas de trabajo de las CPU. Así pues, el uso del proyecto podría mejorar la huella ecológica.

7.2.2 Dimensión Económica

Parecido al anterior apartado, el impacto principalmente proviene del hardware extra. El Optane de Intel es reciente, por lo que no es barato, y de momento está orientado para servidores, lo cual también indica un precio elevado.

Por lo que se refiere a recursos humanos, el proyecto requeriría de un mantenimiento parecido al de una base de datos de la misma magnitud.

7.2.3 Dimensión Social

En el apartado **1.3. Agentes implicados** ya se especifican los beneficiados de este proyecto. Incluso si los resultados no hubiesen indicado una mejora de rendimiento, resultaría útil desde un punto de vista de investigación, ya que utiliza herramientas jóvenes todavía poco documentadas. Con una mejora, podría influenciar a proyectos cercanos del mismo departamento que también trabajan sobre bases de datos.

Tanto el Optane de Intel como Arrow se ha demostrado que pueden ser incluidos en una base de datos. Así pues, está investigación también puede interesarle a compañías especializadas al almacenamiento y gestión de datos. En el futuro, estos centros podrán referirse a este trabajo y tomarlo como ejemplo.

7.3 Riesgos

Incluye los riesgos inherentes al propio proyecto durante toda su construcción, vida útil y desmantelamiento.

7.3.1 Dimensión Ambiental

Actualmente, mucho hardware informático no dispone de puntos de reciclaje. El Optane de Intel no es la excepción, y muy probablemente los DIMMs en desuso acaben en fosas de basura. Si la vida útil del proyecto que utilice esta tecnología es lo suficientemente larga, el impacto ambiental ya no surge de un riesgo atípico, si no de un proceso iterativo en el que el hardware tiene que renovarse.

Globalmente, este proyecto puede aumentar la huella ecológica para mal. Sin embargo, es cierto que, como se ha expuesto anteriormente, el aumento de rendimiento puede compensar la repercusión ambiental del hardware necesario extra.

7.3.2 Dimensión Económica

En el capítulo **2.1.4. Riesgos críticos** ya se han definido los posibles riesgos que pueden suceder durante la fase de producción. En general, los más graves son en los que no tengo responsabilidad alguna. Por ejemplo, que el equipo dentro de Computer Science se quede sin financiación económica, que se rompa el acuerdo entre BSC e Intel o que la máquina Apache Pass del BSC sufra un error grave como un daño físico provocado por un incendio o parecido.

Por desgracia, estos casos serían irreversibles para este proyecto y no se podría haber tomado ninguna prevención. Por otro lado, las probabilidades de estos errores eran mínimas y al final no han acabado ocurriendo.

7.3.3 Dimensión Social

A nivel social este proyecto no supone ningún riesgo. Respecto a los agentes implicados todos tienen las de ganar. En el anterior apartado, ligado con la parte económica, se ha especificado la posible ruptura entre BSC e Intel.

Por otro lado, el impacto que pudiese suponer este proyecto no tendría daños colaterales. A pesar de trabajar con tecnologías nuevas, el trabajo no dispone de la magnitud suficiente como para que tenga una repercusión más allá de los agentes implicados.

7.4 Conclusiones de la sostenibilidad

Como todo proyecto informático, el consumo eléctrico parece ser el primero impacto en sostenibilidad que se nos viene a la cabeza. Asimismo, su impacto social es innegable también, ya que diversos agentes se ven implicados en el trabajo, tanto personales como empresariales. Sin embargo, como el principal objetivo del proyecto es ganar rendimiento en bases de datos, con esta ganancia se podría reducir el consumo de estas aplicaciones.

Por otro lado, tenemos el extra hardware requerido si se hace uso de la tecnología de Intel. Aunque pretende sustituir los discos tradicionales HDD y SSD, de momento el Optane es demasiado caro y los dispositivos no proporcionan tanta capacidad de almacenamiento para las bases de datos. Así pues, estos dispositivos sí que afectarán a la sostenibilidad. Primero de todo tenemos la producción de este hardware, el cual son recursos extras y, probablemente, un impacto social en los lugares, mayoritariamente tercermundistas, donde se extraen los minerales para su fabricación. Segundo, tener más hardware requiere de más espacio, más consumo, más cableado, más hardware, más mantenimiento, etc. Por último, tenemos su desuso, que también deja una gran huella ambiental puesto que para estos

dispositivos todavía no se pueden reciclar eficientemente. Así pues, aunque de momento sea hardware complementario y opcional, su impacto en la sostenibilidad es evidentemente presente.

8 Competencias técnicas

En la inscripción del proyecto se especificaron las competencias técnicas que tendría asociado el proyecto. Redactando la memoria, estas competencias no han cambiado y con menor o mayor medida se han alcanzado:

- **CEC2.1** - Analizar, evaluar, seleccionar y configurar plataformas hardware para el desarrollo y ejecución de aplicaciones y servicios informáticos.
- **CEC2.2** - Programar considerando la arquitectura hardware, tanto en ensamblador como en alto nivel.

El Optane de Intel es el que se está poniendo a prueba. Aunque Apache Arrow, un software, ha ocupado gran parte del trabajo, todo se ha programado consciente del nuevo hardware. La primera competencia refleja esto, en el que se ha adaptado la base de datos Cassandra y los datos columnares de Arrow al nuevo dispositivo no volátil el cual gestionará el almacenamiento. Con esto, se ha comparado y evaluado con un Cassandra con disco HDD y se han extraídos los resultados pertinentes. La segunda se caracteriza por tratar el dispositivo de Intel como un anexo de la jerarquía de memoria tradicional (capítulo **4.2.2. Intel's Optane Memory**). A pesar de que sólo se ha trabajado en lenguaje de alto nivel, esta memoria no es transparente al programador, por lo que el código se ha tenido que adaptar considerando el hardware.

9 Referencias

[1] "Apache Arrow", Apache Arrow, 2019. [Online]. Available: https://arrow.apache.org. [Accessed: 24 Sep 2019].

[2] "The Cassandra Open Source Project", *Openhub.net*. [Online]. Available: https://www.openhub.net/p/cassandra/analyses/latest/languages_summary. [Accessed: 24 Sep 2019].

[3] "Nodetool Usage", *Cassandra.apache.org*. [Online]. Available: http://cassandra.apache.org/doc/latest/tools/nodetool/nodetool.html. [Accessed: 24 Sep 2019].

[4] "cqlsh: the CQL shell", *Cassandra.apache.org*. [Online]. Available: http://cassandra.apache.org/doc/latest/tools/cqlsh.html. [Accessed: 24 Sep 2019].

[5] "The cassandra-stress tool", *Datastax.com*, 2019. [Online]. Available: https://docs.datastax.com/en/archived/cassandra/3.0/cassandra/tools/toolsCStress.html. [Accessed: 13 Jan 2020].

[6] Kalita, C., Barua, G. and Sehgal, P. (2018). *DurableFS: A File System for Persistent Memory*. [ebook]. Available at: https://arxiv.org/ftp/arxiv/papers/1811/1811.00757.pdf [Accessed 24 Sep. 2019].

[7] "Intel® Optane™ DC Persistent Memory Partner: Oracle", *Intel.com*, 2019. [Online]. Available: https://www.intel.com/content/www/us/en/products/docs/memory-storage/optane-persistent-memory/oracle-partner-video.html. [Accessed: 24 Sep 2019].

[8] "Apache Cassandra™ Version History | Datastax", *Datastax.com*, 2018. [Online]. Available: https://www.datastax.com/resources/infographic/apache-cassandratm-version-history. [Accessed: 24 Sep 2019].

[9] "Apache Cassandra™ | Database Software | Datastax", *Datastax.com*. [Online]. Available: https://www.datastax.com/products/apache-cassandra. [Accessed: 24 Sep 2019].

[10] "Apache Cassandra", *Cassandra.apache.org*. [Online]. Available: http://cassandra.apache.org/. [Accessed: 24 Sep 2019].

[11] Jim, P.E. and Brian, Z.Z. (2017). *Fast Access to Columnar, Hierarchically Nested Data via Code Transformation*. [ebook] Available at: https://arxiv.org/pdf/1708.08319.pdf [Accessed: 14 Oct 2019]

[12] “Mirror of Apache Cassandra; Branch: 13981_llpl_engine”, Github.com. [Online]. Available: https://github.com/shyla226/cassandra/tree/13981_llpl_engine. [Accessed: 14 Oct

2019].

[13] "Java Driver for Apache Cassandra 3.2 | Paging", *Datastax.com*, 2019. [Online]. Available: https://docs.datastax.com/en/developer/java-driver/3.2/manual/paging/. [Accessed: 13 Jan 2020].

[14] "For Students: Free Professional Developer Tools by JetBrains", JetBrains.com. [Online]. Aviable: https://www.jetbrains.com/student/. [Accessed: 7 Oct 2019].